
FLUID MODELLING OF MAGNETOPLASMADYNAMIC THRUSTERS



ESCUELA POLITÉCNICA SUPERIOR
DEPARTMENT OF BIOENGINEERING AND AEROSPACE ENGINEERING
MADRID, SEPTEMBER 2014

BACHELOR'S DEGREE IN AEROSPACE ENGINEERING
BACHELOR THESIS

AUTHOR:
ÁLVARO SÁNCHEZ VILLAR

ADVISOR:
DR. EDUARDO AHEDO GALILEA

Acknowledgements

Realizar este proyecto como tantas otras metas no hubiese sido posible sin el incondicional apoyo que he recibido constantemente de mi madre y mi padre, así como de toda mi familia. Por otro lado, he de agradecer a mis compañeros de carrera todo el apoyo que me han dado así como haber creído siempre en mí. Compañeros y amigos como Nicolás, Maria F., Maria C., Eric, Álvaro, Enrique, Santiago o Pablo. Mi especial mención a mi amigo Miguel, el cual ha estado siempre ahí, siempre. A mis amigos del Altair, Rodrigo, Jose, Manuel, Álvaro, Jacobo y Pablo.

No querría olvidarme de aquellos profesores que me concedieron el don de la motivación por transmitirme su enorme interés por la ciencia y la ingeniería. Desde profesores Jose Miguel, Ángel Blasco hasta Manuel, Manolo o mi tutor, Eduardo. Agradecimientos en especial a este último, por toda su paciencia y las horas dedicadas así como el incalculable valor de su consejo y directrices en la realización de este proyecto.

Abstract

This document aims to describe the state of the art and some basic physics of self-field Magnetoplasmadynamic Thrusters (MPDT), which constitute a very interesting technology for high power (Megawatt range) space propulsion. The Thesis starts discussing the frame and expectations for electric space propulsion and the different categories of devices. Then, the operational principles and the literature background of MPDTs are briefly reviewed.

The central part of the Thesis is devoted to understand and model the plasma production and acceleration processes in the MPDT chamber. A two-step progress is followed, materialized in two mathematical models. The first, the simplest one, describes the axial acceleration of a fully-ionized, hypersonic plasma beam. This allows getting familiar with main physical phenomena, dimensionless parameters, performances, parametric investigation, and mathematical methods to deal with a boundary problem of a set of algebraic-differential equations.

The second model adds the neutral gas population and the multiple collisional processes taking place in the discharge, and aims to reproduce the whole plasma production stage in addition to the accelerating one. The mathematical formulation of the model is much more complex and rigorous, and the numerical integration is challenging because of the presence of singular/sonic transitions at intermediate points, the presence of terms of very different orders of magnitude, and the stiffness of boundary condition fulfilment with respect to parametric variations. A Matlab code has been totally built for this model. The comparison of the two models reveals excellent similarity and validates the two-step method followed. Further work should deal with radial dynamics and total energy balance considerations.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Document structure	3
1.3	Objectives	3
2	Electric Propulsion	5
2.1	The motivation of Electric Propulsion	5
2.2	The field of Electric Propulsion	7
2.3	Types of Electric Propulsion	9
3	MPD Thrusters	17
3.1	Operational principles	18
3.1.1	Self-Magnetic field	18
3.1.2	Lorentz force: simplified momentum equation	19
3.1.3	Performance: Mechanical work	22
3.2	Ideal SF-MPDT	23
3.3	MPDT Main Parameters	25
3.4	Onset phenomena	26
3.5	Prototypes	27

3.6	Applied-Field MPDT	31
3.7	Future challenges and research	32
4	Acceleration model	35
4.1	Assumptions	35
4.2	Model description	36
4.2.1	Theoretical basis	36
4.2.2	Dimensional analysis	39
4.2.3	Matlab code	39
4.2.4	Notes on thruster performance	40
4.3	Results: Spatial response.	40
4.4	Results: Nondimensional performance maps	43
4.5	Results: Parametric study	46
5	Complete Axial Model	51
5.1	Assumptions	51
5.2	Model development	52
5.2.1	1 st set of equations	52
5.2.2	Simplification to a 1D model	53
5.2.3	MPD: axial model with ionization	56
5.2.4	Determination of conditions close to S	60
5.2.5	Nondimesionalization	62
5.3	Complete Axial Model: final version	64
5.4	The Matlab Code	65
5.5	Results	68
5.5.1	Case analysis	69

<i>CONTENTS</i>	vii
5.5.2 Operational performance and subsonic region	74
5.6 Comparison with acceleration model	76
6 Conclusions	79
A Budget	83
B Nomenclature	85
C Matlab Codes	89

List of Tables

2.1	Characteristic velocity increments for planetary missions [14].	6
2.2	Space propulsion systems characteristic specific impulse ranges [14].	7
2.3	Typical performance parameters for existing Electric Propulsion systems (Several references).	15
4.1	Typical values of $\ln\Lambda$	37
4.2	Princeton Benchmark Geometrical data.	40
4.3	Operational parameters selected	40
4.4	Engine performance parameters	41
4.5	Flow characteristic parameters	41
5.1	Engine operational parameters for all cases studied	69
5.2	Sonic conditions for 5 eV case.	70
5.3	Sonic conditions for 4 eV case.	71
5.4	Sonic conditions for 3 eV case.. . . .	72
5.5	Sonic conditions for 2 eV case.	73
5.6	Performances and obtained parameters of the cases studied	74
5.7	Debye lengths for the practical cases	76
A.1	Project costs	84

List of Figures

1.1	Diagram of the approximated operational regions for EP systems as a function of the power required and the specific impulse [21].	2
2.1	Sketch of a resistojet and its main components [11].	10
2.2	Image of a test on a Ion thruster [1].	12
2.3	Illustration of the SMART-1 Hall thruster [2].	14
2.4	Representation of a MPDT [3].	14
3.1	Simplified sketch of self-field MPD thruster (SF-MPDT) [5].	18
3.2	EPEX MPD arcjet thruster tested in space [22].	27
3.3	EPEX MPD arcjet thruster being tested in vacuum chamber[22].	27
3.4	Schematic of the Princeton full-scale benchmark thruster. The dimensions are $r_c = 0.95$ cm, $r_a = 5.1$ cm, $r_{ao} = 9.3$ cm, $r_{ch} = 6.4$ cm, $t_a = 0.95$ cm and $l_c = 10$ cm.[10]	28
3.5	Schematic of the DT2 MPD thruster [6].	29
3.6	Schematic of the ZT3 MPD thruster [6].	30
3.7	Schematic of KeRC MPD thruster [13].	31
3.8	Schematic of an AF-MPDT [6].	32

4.1	Evolution of the nondimensional variables along the longitudinal axis of the thruster	42
4.2	Spatial response of main variables	43
4.3	Map of the behaviour of the flow Electric field with the magnetic reynolds number	44
4.4	Error of the approximation formula	44
4.5	Efficiency map	45
4.6	Parametric dependance w.r.t. current intensity	46
4.7	Parametric dependance w.r.t. mass flow rate	47
4.8	Parametric dependance w.r.t. T_e at 23000 A	48
4.9	Efficiency as a function of the power consumption and the mass flow rate	48
4.10	Power delivered as a function of the mass flow rate showing the lines of constant efficiency η_p	49
5.1	Collisional rates dependances as a function of T_e for Ar species.	68
5.2	Complete solution for 5 eV case.	70
5.3	Subsonic region for 5 eV case.	70
5.4	Complete solution for 4 eV case.	71
5.5	Subsonic region for 4 eV case.	71
5.6	Complete solution for 3 eV case.	72
5.7	Subsonic region for 3 eV case.	72
5.8	Complete solution for 2 eV case.	73
5.9	Subsonic region for 2 eV case.	73

Chapter 1

Introduction

The objective of this Bachelor Thesis is to review the existing literature on MPDTs and to analyze the flow and the performances of a MPDT for space propulsion, by developing a mathematical model and implementing a computational code.

1.1 Motivation

Electric propulsion (EP) is a research area with a promising future. Conventional chemical thrusters have been overcome in specific missions concerning space exploration. The development of deep-space exploration has been possible by the great improvements made on this technology. Overall, EP thrusters own an incredibly advantageous characteristic. They enjoy the capacity to increase their specific impulse in more than one order of magnitude with respect to that provided by the most advanced chemical thrusters [8].

Several EP technologies have been exhaustively studied as the arcjets, resistojets, pulsed-plasma thrusters, ion thrusters, Hall thrusters, etc. Nevertheless, less effort has been dedicated to high power EP thrusters. These engines are efficient at powers in the range of hundreds of kilowatts even in the Megawatt range. This power range cannot be provided by the state of the art of powerplants available for space. Therefore, the study of the physical phenomena occurring in these engines by computational modelling, can avoid considerable costs either in energy or in time.

Among these engines, stands out the Magnetoplasmadynamic Plasma thruster. This engine is capable to provide specific impulses up to 5000 s and has a significant advantage versus its competitors. It can provide thrust densities that overcome the typical values for its opponents in two orders of magnitude. Then, these thrusters are seen as a compact, powerful and long-range capable EP system. This way, the MPDT could fulfill the requirements of long-range high-cargo missions, decreasing also the duration of trips and enabling the use of EP systems for more impulsive operations. Missions as impulsive orbit transfers or high-cargo transport to outer planets are some of the most relevant purposes of this technology. The power level that these thrusters require, entails the necessity of deep improvement of powerplant technology.

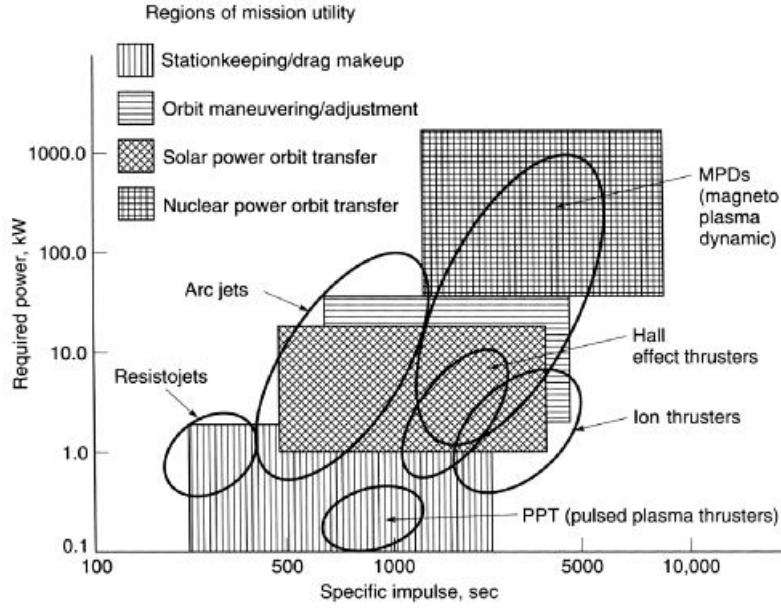


Figure 1.1: Diagram of the approximated operational regions for EP systems as a function of the power required and the specific impulse [21].

Below 200 kW, the self induced magnetic field is insufficient for the efficient operation of the thruster. Therefore, the self-field MPD thrusters (SF-MPDT) are discarded as low power technology. An alternative in the range of 20-200 kW are the applied-field MPD thrusters (AF-MPDT), in which as their name points out, an external magnetic field is applied. However the physical phenomena present in these thrusters is beyond the scope of this Bachelor Thesis. This document focuses on the study of SF-MPD thrusters.

High power is needed to obtain the required currents for the thruster to operate in the electromagnetic regime where $\beta < 1$. This parameter as other important parameters governing the operation of MPD thrusters will be explained in the following chapters of this document. Operating at such high currents causes severe damage in the cathode from surface erosion. This fact together with the high power requirement have slowed down the development of these thrusters. Their viability could mean an inflexion point in the field of space exploration [8].

The main intention of this document is to revisit the main physics and dynamics of these engines and to perform an analysis of these engines with simplified models. Two models are discussed throughout the document.

The first model, based on previous work done by Prof. Manuel Martinez Sánchez [20], studies the acceleration dynamics of a MPDT having only the Lorentz Force present in the momentum equation. Using an iterative code, the mathematical problem is solved fulfilling the boundary conditions. The code uses a model that has been previously nondimensionalized. It also obtains a simple relationship between the

thruster propulsive efficiency η_t and the magnetic Reynolds number Rm . Furthermore, it is able to calculate a first estimate of the performance parameters of the thruster.

The second model studies the complete axial acceleration process, taking into account the Lorentz Force, the pressure gradient, and also the ionization and other collisional processes. Typically the propellant is not ionized before entering the chamber. In fact, plasma ionization can occur in a large region throughout the engine. This code assesses the behavior of this process as well and seeks for ionization spatial scales. Moreover plasma pressure and therefore subsonic-supersonic transition occurs in the acceleration process, and are important mainly below 200 kW of power delivered, as the total acceleration obtained in that range is small and both subsonic and supersonic regions become comparable. This term adds a singularity in the mathematical problem, that is overcome by an rigorous mathematical manipulation of the model equations. The code returns the spatial response of the different problem variables as well as the main thruster performance parameters.

1.2 Document structure

This document presents the socioeconomical context of EP field and the motivation of its research and development in Chapter 2. Also in Chapter 2, the field of electric propulsion is discussed as well as the types of EP thrusters giving some examples of them. In Chapter 3, the state of the art of MPD thrusters is described, by introducing the main physical principles and phenomena. Some SF-MPDT prototypes of interest are described as well. In Chapter 4, the Acceleration Model for a fully-ionized plasma is described, from the development of the model to the final results obtained. Chapter 5 details the second model and follows a similar scheme to the first model. It depicts the model development, describes the code operation and represents the results obtained. At the of Chapter 5, a comparison of both models is performed. Finally Chapter 6 outlines the conclusions reached throughout the development of this project. At the end of the document, additional information is attached in appendices, as the project budget, the document nomenclature or the matlab codes implemented for both models.

1.3 Objectives

This section outlines which are the main purposes of this Bachelor's Thesis.

The main objective of this Bachelor's Thesis is to be able to model and represent the operation of a MPDT. The models developed try to perform this task, one only considering the Lorentz force in the acceleration process, and the last model, taking into account the complete axial acceleration process with all physical phenomena included in the study. Also the physical phenomena of MPD thrusters is to be revisited as well as a bibliographic study to seek for self-field MPD prototypes already manufactured. Therefore the objectives of the project are:

1. Understanding the field of MPD thrusters.
2. Obtaining basic understanding of the main physical MPD phenomena.
3. Develop an acceleration model without including ionization.
4. Develop a model that includes all terms in the one dimensional momentum equation for partially ionized plasma.

Notice that the crucial part of this Bachelor's Thesis is the modeling process of MPD thrusters developed and implemented in MATLAB software. Not only both models will have to show the spatial response of the main variables governed, but also to obtain operational performance parameters. Also the results obtained from these models will be analyzed with the use of theoretical means to extract global conclusions on the MPDT operation.

Chapter 2

Electric Propulsion

Electric propulsion is nowadays the most efficient space propulsion system far from high gravitational fields, overcoming conventional chemical rockets.

This technology has been successfully used in the space industry and has enabled the development of deep-space missions as the Dawn Mission. In this mission an ion engine propelled a spacecraft exploring the asteroid belt present between Mars and Jupiter.

Electric propulsion appears in the 1990's as one of the alternatives to chemical propulsion in the role of propulsion systems for space missions [8].

In fact, electric propulsion is a technology that was conceived by Robbert H. Goddard in 1906. In the next decades more rocket scientists developed the first theories concerning electric propulsion, as professor Herman Oberth did in 1929. No great efforts were made in this technology until practical nuclear-fission power sources and solar panels reached the required power values for EP thrusters [14]. EP has been studied since the post-Second World War era and nowadays is a growing branch of space propulsion research and development.

2.1 The motivation of Electric Propulsion

Propulsion systems are necessary to apply a vectoring force on the mass we want to translate. Due to Newton's 2nd law that states that the motivation of the acceleration of a mass is only provoked by the resultant force of all the forces applied to it, it can be claimed that for the same acceleration, the higher the mass the higher the force is needed to apply on the object.

$$\frac{d\mathbf{P}}{dt} = \frac{dm\mathbf{v}}{dt} = \sum \mathbf{F}$$

As a basic introduction to the topic, let us assume a rocket moving in a gravitational field. The equation of motion of that rocket is defined by the following vectorial differential equation:

$$\dot{m}\mathbf{v} = \dot{m}\mathbf{u}_E + \mathbf{F}_{ext} \quad (2.1)$$

where \mathbf{v} : acceleration vector of the rocket

\dot{m} : mass flow rate of the engine or rate of change of the rocket mass.

\mathbf{u}_E : exhaust velocity relative to the rocket

\mathbf{F}_{ext} : external forces (normally gravitational drag and in presence of air also aerodynamic drag).

For the sake of simplicity let us assume that the problem is one-dimensional and that there is no gravitational drag or aerodynamical drag, so $\mathbf{F}_{ext} = \mathbf{0}$. The term on the right is the thrust ($\mathbf{T} = \dot{m}\mathbf{u}_E$) of the rocket.

The exit velocity u_E is sometimes called specific impulse but conventionally is defined as $I_{sp} = \frac{u_E}{g_0}$.

Integrating (2.1) for an initial instant t_0 to a final instant t_1 , it can be obtained one of the most important equations of the rocket science, the so-called Rocket Equation that was described for the first time by Konstantin Tsoikolsky.

$$\Delta v = g_0 I_{sp} \ln \frac{m_0}{m_1} \quad (2.2)$$

where m_0 is the mass at the start of the mission and m_1 is the mass at the end of it.

Every single space mission has its own required Δv to fulfill the mission. Some of the characteristic values of Δv for planetary transfer missions are showed in table 2.1.

Mission	Maneuver type	$\Delta v(m/s)$
Escape from Earth Surface	Impulsive	1.12×10^4
Escape from 300-mile orbit	Impulsive	3.15×10^3
Escape from 300-mile orbit	Gentle spiral	7.59×10^3
Earth orbit to Mars orbit and return	Typically impulsive	1.4×10^4
Earth surface to mars surface and return	Typically impulsive	3.4×10^4
Earth orbit to Venus orbit and return	Typically impulsive	1.6×10^4
Earth orbit to Mercury orbit and return	Typically impulsive	3.1×10^4
Earth orbit to Jupiter orbit and return	Typically impulsive	6.4×10^4
Earth orbit to Saturn orbit and return	Typically impulsive	1.1×10^5

Table 2.1: Characteristic velocity increments for planetary missions [14].

This equation is the main motivation of the electric propulsion technology. The I_{sp} becomes a relevant parameter in the design process of a mission because of its impact on the rocket equation. EP thrusters have a great specific impulse, much larger than that of the chemical propulsion systems as can be seen in table 2.2.

Engine	Δv (m/s)
Liquid monopropellant thrusters	1700 – 2900
Solid propellants thrusters	2100 – 3200
Liquid bipropellants thrusters	2900 – 4500
Exotic bipropellants and tripropellants thrusters	4000 – 6000
Electrothermal thrusters	1500 – 15000
Electrostatic thrusters	15000 – 120000
Electromagnetic	3000 – 60000

Table 2.2: Space propulsion systems characteristic specific impulse ranges [14].

As a consequence, two important advantages appear with the development of electric propulsion. Firstly, missions thrust in space by EP systems could decrease the costs of the mission since they can provide the same Δv with lower amount of total mass expelled. Secondly, EP allows to advance to greater distances with the same propellant mass, even to places that haven't been explored before, opening the gates of deep space exploration.

2.2 The field of Electric Propulsion

All the comments made on EP thrusters suggest that electric propulsion is a technology that prevails over chemical propulsion. However, the denial to that statement comes clearer when looking at other figures of merit of thrusters as the power required and impelled thrust.

Concerning any type of thruster another figure of merit is the thruster efficiency η_t . It basically represents the proportion of energy used to thrust the vehicle from the total amount of energy available.

$$\eta_t = \frac{T^2}{2\dot{m}P_a} = \frac{Tg_0I_{sp}}{2P_a} \quad (2.3)$$

where P_a is the total power available[4].

The thruster efficiency is an indicator of the thruster competitiveness. For that purpose, energy losses should be diminished as far as possible. However this type of technology has an intrinsic loss due to ionization. The gas must be ionized in order to be accelerated. Propellants with minimum specific ionization energy are optimum for these thrusters.

Moreover, cooling is an issue in space, specially for these engines that operate at high powers with respect to chemical thrusters. Undesired heating could deteriorate the thruster performance and even damage the thruster.

In addition to that, P_a is limited by the existing technology. At the moment the dominant power source for space vehicles is solar power. Solar panels are able to provide power values up to 30 kW. This technology is far from the one needed to operate

some of the EP thrusters that are efficient in the Megawatt range, as is the case of the MPDT. The required powerplant mass that limits the available power is an important figure to decide which engine should propel the spacecraft. Powerplants design seeks for high power-per-unit-mass values (α).

As P_a is such a limiting figure, at this point it is possible to notice that the specific impulse has a direct effect on the thrust for a given efficiency and available power (2.3). This means that a high specific impulse of these thrusters limit them to have low thrust and to operate during large periods of time to obtain the same Δv . Therefore, electric propulsion cannot govern the same type of missions as chemical propulsion. For instance, launch missions, that require high thrusts to scape from Earth's gravitational field, or highly impulsive maneuvers.

Whilst in chemical thrusters the objective is to maximize the specific impulse by a selecting the appropriate propellant, exhaustive design of the nozzle and combustion chambers, cooling systems, etc, in electric thrusters the I_{sp} must be optimized. The reason is that, as it has been explained in (2.3), the thrust obtained from a device that has a too high I_{sp} is considerably low. Therefore, the specific thrust per unit of power is decreased and there is a strong penalty on the payload as the powerplant mass could be too high.

However, there is an optimum specific impulse for which payload fraction is maximum for a combination of design and mission parameters ($\eta_t, \alpha, \Delta t, \Delta V$). The deduction of the formula is explained hereinafter.

The mass of the vehicle can be divided in different contributions:

$$M_0 = M_{STR} + M_{PL} + M_{PPS} + M_P \quad (2.4)$$

M_0 : Vehicle Mass

M_{STR} : Structural Mass.

M_{PL} : Payload Mass.

M_{PPS} : Powerplant Mass.

M_P : Propellant Mass.

An extremely important part of the design of an spacecraft is selection of the powerplant. The mass per unit of power supplied by a powerplant is a critical parameter (α). It is described as:

$$\alpha = \frac{M_{PPS}}{P_a} \quad (2.5)$$

For electric propulsion, powerplants can weight as much as a 30% of the total mass of the vehicle (M_0). Therefore, the power provided and M_0 are normally compared by the following ratio:

$$\frac{P_a}{M_0} = \frac{1}{\alpha} \frac{M_{PPS}}{M_0} \quad (2.6)$$

Assuming that the mass flow rate of the vehicle is constant and that the time of operation is called Δt the following expression of the power can be deduced by using (2.3):

$$P_a = \frac{M_P}{\Delta t} \frac{(g_0 I_{sp})^2}{2\eta_p} \quad (2.7)$$

Returning to (2.4) and plugging in (2.2), (2.5), (2.6) and (2.7), this final expression is obtained:

$$\frac{M_{PL} + M_{STR}}{M_0} = 1 - \frac{M_P}{M_0} - \frac{M_{PPS}}{M_0} = \exp\left(\frac{\Delta V}{I_{sp} g_0}\right) - \frac{(I_{sp} g_0)^2}{V_{ch}^2} \left(1 - \exp\left(\frac{\Delta V}{I_{sp} g_0}\right)\right) \quad (2.8)$$

Notice that V_{ch} is the characteristic speed, defined as:

$$V_{ch} = \sqrt{\frac{2\eta_t \Delta t}{\alpha}} \quad (2.9)$$

For (2.8) there is an optimum $I_{sp} = I_{sp}^*$ for which the payload-vehicle mass fraction is optimized. The structural mass normally is included as payload in this analysis but has been outlined for the sake of accuracy.

For the analysis followed this value is approximately:

$$I_{sp}^* \approx \frac{V_{ch}}{g_0} - \frac{\Delta V}{2g_0} \quad (2.10)$$

Therefore $I_{sp}^* = f(\alpha, \Delta t, \eta_t, \Delta V)$.

For instance, consider a LEO-GEO (Low-Earth-Orbit Geostationary) mission with $\alpha = 0.02 \text{ kg/W}$, $\Delta t = 4.66 \times 10^6 \text{ s}$, $\eta_t = 0.7$, and $\Delta V = 5700 \text{ m/s}$ would mean that $V_{ch} \approx 18000 \text{ m/s}$ and $I_{sp}^* \approx 1600 \text{ s}$. In the case of a mission to Mars with $\alpha = 0.02 \text{ kg/W}$, $\Delta t = 3.17 \times 10^7 \text{ s}$ (one year), $\eta_t = 0.7$, and $\Delta V = 15000 \text{ m/s}$, the resulting characteristic velocity and optimum specific impulse would be $V_{ch} \approx 47000 \text{ m/s}$ and $I_{sp}^* \approx 4000 \text{ s}$.

2.3 Types of Electric Propulsion

This section presents a brief introduction to the most important types of Electric Propulsion systems. Plasma thrusters can be classified according to the mechanisms for plasma production and for plasma acceleration.

Plasma production and heating are obtained by direct-current (DC) biased electrodes or alternate current (AC) antennas, radiating mainly in the radiofrequency

(RF) or microwave domain. The direct current mechanism is used in most veteran designs, but suffer from electrode erosion. The alternate current devices must care about electromagnetic interferences and inefficiencies in the wave energy conversion and transmission.

Plasma thrusters can also be clasified by the different acceleration mechanisms or momentum exchange sources that appear in the momentum equation for a fully ionized plasma :

$$\nabla \cdot \rho \mathbf{u} \mathbf{u} = \epsilon_0 (\nabla \cdot \mathbf{E}) \mathbf{E} + \mathbf{j} \times \mathbf{B} - \nabla \cdot \bar{\bar{P}} \quad (2.11)$$

1. *Pressure force* : $(\nabla \cdot \bar{\bar{P}})$

This type of thrusters is also called **Electothermal Propulsion**. It comprises all different examples of thrusters developed that use electricity to heat the propellant. Then fluid-dynamic expansion takes place changing the thermal energy to kinetic energy and applying a reactive force on the thruster walls. Commonly, convergent-divergent nozzles are used. Two types of thrusters serve as reference for this type of acceleration mechanism based on the state of the mass that is accelerated. On one hand, the engines that use gaseous state mass as propellant (Resistojets). On the other hand, those thrusters that use plasma state mass as propellant (Arcjets). Resistojets use a solid surface as a chamber wall or a heater coil to heat the propellant.

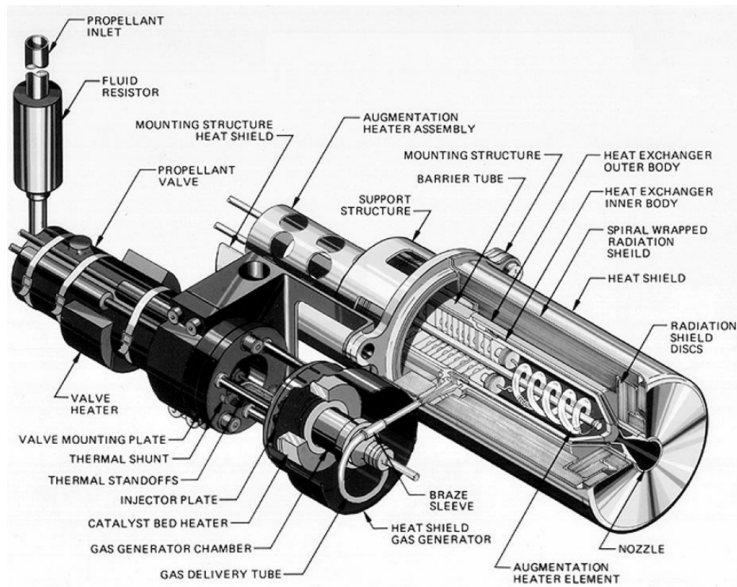


Figure 2.1: Sketch of a resistojet and its main components [11].

Arcjets use the Joule effect to heat the propellant by driving an electric arc through it. Proper design of the current arc geometry is necessary to obtain

tolerance to large radial temperature gradients.

Inductively or radiatively heated devices (IRH) appear as an alternative to arcjets as arc heating diminishes considerably the operational life of thrusters [4],[8],[11],[17].

Engines that accelerate fluid-dynamically the flow by expansion can be studied by the 1D entropy equation for the flow in the engine. Basically the velocity at the exit of the engine depends on the heat capacity of the propellant and the chamber temperature.

$$u_E \approx 2\sqrt{C_p T_c}$$

Therefore, the objective propellant will be one with high heat capacity and also the chamber temperature must be increased as much as possible. As a consequence, the specific impulse of thrusters using this acceleration mechanism is strongly limited by the presence of a figure of merit that is difficult to further increase, the chamber temperature.

2. Electrostatic force : $(\epsilon_0(\nabla \cdot \mathbf{E})\mathbf{E})$

The thermal limitations on maximum attainable exhaust speeds and lifetimes of electrothermal engines suggest another solution to the acceleration process to take place. In fact, the other terms appearing in the momentum equation for the plasma are body forces. The electrostatic force is one of the simplest acceleration mechanisms and the results of the technology developed are incredibly profitable. The propulsive systems governed by this acceleration process are comprised in **Electrostatic Propulsion**.

The main example of electrostatic propulsion is the **Ion Thruster** or **Ion Engine**. The design of the ion engine reveals a clearly distinguishable structure. It is composed by three main parts: the ionization chamber, the grid accelerator and the neutralizer. The ionization chamber is cylindrically shaped with an axial cathode that emits electrons and a surrounding anode shell. In the ionization chamber the propellant is bombarded with electrons commonly confined using permeating azimuthal or radial magnetic field, designed to optimize the ionization. Then the accelerator grid is based on two or three grids that produce a voltage bias which induces an electric field, and subsequently, accelerates the plasma. The grid geometry is designed to minimize the ion beam impingement on the grid which is a main factor reducing lifetime of this class of thrusters. Finally, the neutralizer is used to produce a free charge plasma beam at just some units of grid spacing from the chamber exhaust.

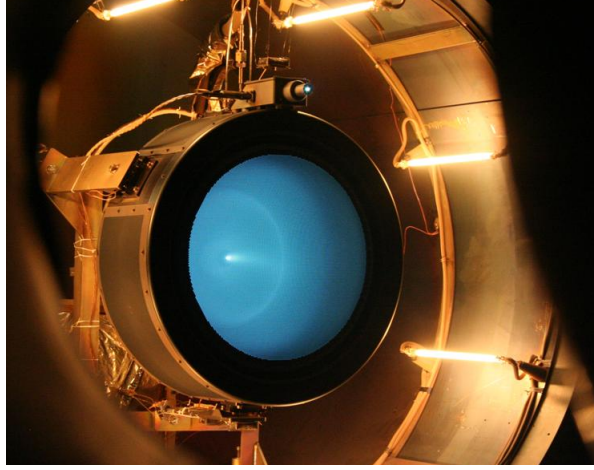


Figure 2.2: Image of a test on a Ion thruster [1].

The penalty of these thrusters is the maximum thrust density that can be attained, which is limited by space-charge distortions of the applied electric field pattern [4],[8],[11].

3. Lorentz force: ($\mathbf{j} \times \mathbf{B}$)

Another acceleration mechanism is the Lorentz force by which an electromagnetic composition of fields produces the flow acceleration. The engines that use this acceleration mechanism to impel thrust are comprised in the so-called **Electromagnetic Propulsion**. The basic principle of operation of electromagnetic thrusters is the interaction of an electric current passing through a propellant stream with a magnetic field in order to produce a body force. Although these engines can deliver exhaust speeds much more elevated than electrothermal devices, the physical phenomena that they present, is the most complicated among the three types of electric propulsion. However, it is enhanceable the fact that these engines normally use plasma state propellant and it is quasineutral (this concept will be further explained in chapter 3). Much higher thrust densities can be attained as they lack of space-charge limitations, an important disadvantage of electrostatic technologies.

Returning to (2.11), in order to compare electrostatic and electromagnetic capabilities, some calculations can be made. For that purpose, let us describe the different acceleration mechanisms as pressures, and seek for numbers to reveal the capacities of each technology concerning thrust densities. Therefore two different new concepts appear: Electrostatic pressure & Magnetic pressure.

On one hand, electrostatic pressure is defined as $p_E = \frac{\epsilon_0 E_a^2}{2}$ where $\epsilon_0 = 8.85 \times 10^{-12}$ F/m, is the vacuum permeability and $E_a = \frac{4V}{3d}$, the field on the surface of the extractor electrode. Knowing that E_a is rarely higher than 2000 V/mm [19], the maximum estimated electrostatic pressure that can be provided to the

plasma is 20 Pa.

On the other hand, magnetic pressure is defined as the pressure provided by the magnetic field and is given by: $p_{magnetic} = \frac{B^2}{2\mu_0}$ where B is the magnetic field modulus and $\mu_0 = 4\pi \times 10^{-7}$ H/m, is the magnetic permeability in vacuum. A 1000 G magnetic field (equivalent to 0.1 T) can be easily applied by using coils or permanent magnets. Therefore the resulting pressure obtained is approximately equal to 8 kPa. Comparing both pressures, the magnetic pressure can be 400 times greater than the maximum limit for electrostatic pressure. Notice that this calculation is qualitative and is not meant to be taken as a quantitative argument.

Electromagnetic engines use an electrically conducting fluid (ionized gas) without appreciable net charge, to drive through an electric current and subjected to some combination of electric and magnetic fields. Both the electric field \mathbf{E} and the magnetic field \mathbf{B} are perpendicular to each other and to the stream velocity \mathbf{u} . Three engines must be enhanced when dealing with electromagnetic thrusters: Hall Thrusters, Helicon Thrusters and Magnetoplasma dynamic Thrusters. Note that there are more examples of electromagnetic devices that are not described in this document as the Pulsed Plasma Thrusters (PPT), Pulse Inductive Thruster (PIT), etc.

Hall thrusters receive that name from Edwin H. Hall that was then a physics graduate student at Johns Hopkins University when he formulated the nowadays called Hall Effect. Hall discovered that when an electric field and magnetic field are applied perpendicular one to each other inside a conductor, an electric current (or the so-called Hall current) appears in a direction perpendicular to both fields [8]. A set of properly disposed magnetic coils produce a radial magnetic field which induces an azimuthal current that flows around the central anode ($T_{hall} = j_{\theta} B_r$). An anode is placed at the entrance of the chamber and a cathode outside the chamber, close to the chamber exit. While the magnetic field provokes the rotation of the electrons around the central insulator, producing the Hall current, the propellant enters the chamber. It is ionized by collisions with the electrons and positive ions are expelled by the potential difference exerted between both electrodes. Therefore, although these engines are considered electromagnetic engines, they could be also classified as electrostatic since the ions acceleration is provided in that way. Furthermore some electrons are dragged from the cathode to neutralize the plasma at the exit.

These engines have resulted to be very beneficial and several missions have taken place such as ESA SMART-1 mission to the moon.

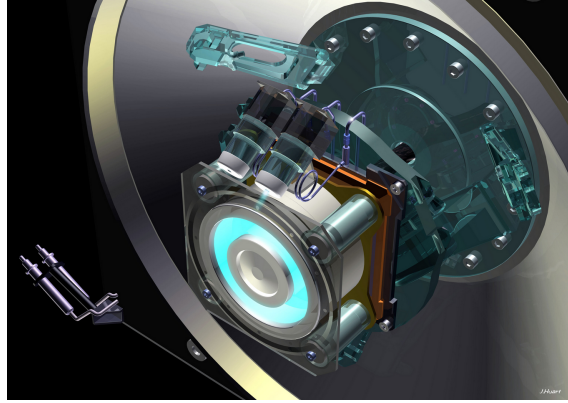


Figure 2.3: Illustration of the SMART-1 Hall thruster [2].

One of the latest of these engines is the **Helicon thruster**. Helicon thruster technology dates back from 10 years ago and is in a very incipient stage with less than a dozen prototypes and active experiments worldwide. The typical thruster architecture is as simple as a dielectric tube (quartz normally) surrounded by solenoids and RF antenna, emitting in the range of 1-25 MHz [4]. This technology is based on helicon antennas that excite the propellant by low frequency electromagnetic waves or also known as helicon waves. These waves efficiently ionize the propellant. The magnetic field confines the plasma from the walls functioning as a magnetic nozzle and also provides permeability of the plasma to helicon waves absorption and propagation.

Finally, the most promising engine for long range high-cargo space missions according to authors as Edgar Y. Choueiri, professor of Princeton University [8], is the **MPDT**. These engines use the same principle of operation of Hall thrusters but the Lorentz force is obtained from a radial current and an azimuthally induced magnetic field ($T_{MPD} = j_r B_\theta$). As this project is focused on the study of these engines, an exhaustive description of them is detailed in Chapter 3 (MPD Thrusters).



Figure 2.4: Representation of a MPDT [3].

Notice that this classification is not the most rigorous since some of the engines described operate with more than one acceleration mechanism. For example, Helicon thrusters can be considered both electrothermal and electromagnetic devices as they use both acceleration mechanisms. Moreover, the I_{sp} and T of Helicon thrusters depends highly on the temperature reached in the chamber.

Nevertheless, there exist EP engines that have not been mentioned as it is not the scope of this Bachelor Thesis.

From an engineering point of view is mandatory to appreciate the differences between systems in the relevant parameters that affect their performance. Concerning thrusters the main parameters that must be taken into account are:

- Thrust T
- Power required P_{req}
- Efficiency η_t
- Specic impulse I_{sp}
- Typical propellants

Table (2.3) compares the existing technologies with an approximate range of the parameters previously enumerated.

<i>Technology</i>	T [mN]	P_{req} [kW]	η_t [%]	I_{sp} [s]	Propellants
Resistojet	250-300	0.5-1.5	65-90	200-350	H ₂ ,NH ₃ ,N ₂ H ₄
Arcjets	200-1000	0.3-30	30-50	400-1000	H ₂ ,NH ₃ ,N ₂ H ₄ ,N ₂
Ion Thruster	0.01-500	1-7	60-80	1500-10000	Ar,Kr,Xe,Bi
Hall Thruster	0.01-2000	1-10	50-55	1500-2000	Ar,Xe
Helicon Thruster	1-1000	0.05-50	10-40	500-2000	Ar,Xe,Kr,N ₂ ,H ₂
SF-MPD Thruster	0.001-50000	100-4000	≤ 40	2000-5000	Ar,Xe,Li,H ₂

Table 2.3: Typical performance parameters for existing Electric Propulsion systems (Several references).

Not all the technologies detailed in table 2.3 are at the same stage of development. Furthermore, some of the thrusters described are still in the research stage.

Chapter 3

MPD Thrusters

Having already discussed and introduced Electric Propulsion, the document focuses on Magnetoplasmadynamic thrusters hereinafter. Main concepts on physical phenomena and examples of built engines are provided in this chapter.

As has been seen the electrothermal technology was limited on its maximum specific impulse by the chamber temperature and the specific heat capacity of the propellant. This limitation provoked the opening of new R&D lines as electrostatic and electromagnetic EP systems. The workhorse of electric propulsion, the Ion Thruster, provides high I_{sp} but the maximum achievable thrust density is strongly limited by space-charge limitations. Electromagnetic engines appear there to solve this issue with the development of Hall thrusters that increase in one order of magnitude the thrust density.

In order to further increase the thrust density, the seek for a different approach of the latter technology is required. In fact, Hall Thrusters cannot operate at very high plasma densities since the highly collisional character of the plasma interrupts the electron movement and so the Hall current required for the acceleration process. The solution is obtained from aligning the electric field with the current used, that is a configuration far less prone to collisional disruptions [8]. The main acceleration mechanism of an MPDT is also the Lorentz force but as explained the configuration of the electric and magnetic field combination is changed. These engines allow for denser plasmas, around $n_e \approx 10^{21} \text{ m}^{-3}$, larger than all other technologies.

The plasma present in an MPDT, is typically cold and highly collisional. The electron temperature (T_e) has been proven to be below 10 eV by several tests on existing thrusters. A reasonable estimate of the range of temperatures is proven to be 2-5 eV [9],[18].

Starting from the thruster architecture, the MPD, can be seen as conceptually simple. A central hollow cathode is surrounded by an anode shell. The propellant, is driven through the cathode and at the tip is expelled to the cathode exit. Thermionic emission of electrons from the cathode is combined with the acceleration of these by the voltage bias applied. These electrons collide with the neutrals, ionizing the propellant. The resulting plasma conducts a current between anode and cathode inducing an

azimuthal magnetic field. This self-induced magnetic field interacts with the electric current, to provide the acceleration of the flow and therefore the thrust [8],[5].

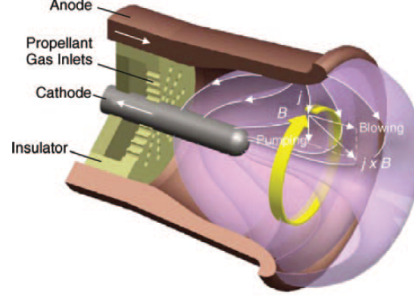


Figure 3.1: Simplified sketch of self-field MPD thruster (SF-MPDT) [5].

In order to deepen our knowledge on these engines let's summarize their basic principles, develop some aclaratory theories as well as describe the most important phenomenological concepts to have awareness of.

3.1 Operational principles

3.1.1 Self-Magnetic field

MPD thrusters can be subdivided in two investigation lines. The line that is studied along this Bachelor Thesis is the Self-field MPD thrusters. These engines operate with the use of an exclusive principle. The magnetic field used to accelerate the plasma is induced by the electric current that appears inside the thruster chamber. The voltage difference between anode and cathode is fed by a power supply. Current is driven through the plasma between anode and cathode and then comes back to the power supply. This process can be seen in the following sketch.

The law that explains the appearance of a self-induced current is the so-called Ampere's Law.

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} \quad (3.1)$$

Also, it can be seen in integral form:

$$\iint_{\Sigma} \mathbf{j} \cdot d\mathbf{S} = \oint_{\Gamma} \frac{\mathbf{B}}{\mu_0} d\Gamma \quad (3.2)$$

Ampere's law states that the total flux of current that goes through a surface Σ is equivalent to the circulation on a curve Γ of the $\frac{\mathbf{B}}{\mu_0}$. Taking a radial cross section of the chamber and integrating the net current it can be observed that there exists an azimuthally induced magnetic field.

3.1.2 Lorentz force: simplified momentum equation

The Lorentz force is the main motivation of the acceleration process in MPD thrusters. Therefore it is important to understand properly the physics behind this theory and the principles that these engines use to operate.

The propellant exposed to high-power electric arcs is typically in a plasma state. At scales that can be compared in order of magnitude with those of the thruster, the plasma behaves as a null net-charge conducting medium. To be more concise, the plasma is quasi-neutral, meaning that the amount of electrons is approximately equal to positively charged ions (cations). Another manner of stating quasineutrality is:

$$|n_e - n_i| \ll n_e \approx n_i = n \quad (3.3)$$

where n_e is the electron density, n_i is the ion density, n is the total plasma density.

The assumption of quasineutral plasma states that, opposite to what happens in electrostatic accelerators, different sign charged species stay together throughout the whole acceleration process.

Quasineutrality is a generalized condition for plasmas that can be seen as a property. The presence of net charge in a local scale is automatically regularized by the field that itself induces, moving the free charged particles to neutralize that charge. The only practical case of local net potential in the plasma appears are potential energies of the order of the electrons thermal energy related to their motion (kinetic energy). From this case arises an interesting characteristic length that arises from equating the electrons thermal energy with their potential energy. This characteristic length is the **Debye Length** λ_D and is defined as the characteristic length for which the thermal energy of electrons is comparable to the potential energy changes inside a plasma. This region is such that $\lambda_D \ll L$ and is called sheath. The sheaths appear commonly at the walls of objects in contact with a plasma and are studied as boundary layers in fluid dynamics. Further comments are made on this parameter at the end on the document, where the results are commented.

Returning to the Lorentz force formulation, it seems that the macroscopical point of view of the process is accurate enough for the purposes of this explanation. As a consequence every quantity studied will refer to an average value of all the particles.

The purpose of this study is the explanation of the acceleration process present in MPD thrusters. For that purpose, the momentum equation must be formulated. For the sake of simplicity and clarity, reasonable assumptions must be made.

- The first assumption has already been exposed in (3.3), the plasma is quasineutral.
- Assume the plasma is composed by only two species, cations of type X^+ with subindex i and electrons e^- with subindex e . Therefore the plasma studied in this analysis is fully ionized.

- Typically viscous terms can be neglected because of their negligible contribution to the cases of interest studied.
- The momentum storage capability of the electrons can be neglected w.r.t. the ion contribution.

Taking into account these assumptions the momentum equation for both species, ions and electrons, can be correspondingly formulated:

$$m_i n \frac{D\mathbf{u}_i}{Dt} = ne (\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \nabla p_i + \mathbf{P}_{ie} \quad (3.4)$$

$$0 = -ne (\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \nabla p_e + \mathbf{P}_{ei} \quad (3.5)$$

m_i is the ion mass

\mathbf{u}_e is the electron velocity and u_i the ion velocity.

\mathbf{E} is the electric field and \mathbf{B} is the magnetic field

p_e is the electron pressure and p_i the ion pressures.

\mathbf{P}_{ei} is momentum gain of electrons due to collisions with ions and \mathbf{P}_{ie} is the opposite.

Notice that the term in the left of (3.4), is the density times the material or convective derivative described in (3.6), which analyzes the change of a quantity following the particle.

$$\frac{D()}{Dt} = \frac{\partial()}{\partial t} + \mathbf{u} \cdot \nabla() \quad (3.6)$$

The first term in the right part of (3.4) is the basic principle of the acceleration process of MPD thrusters, the Lorentz Force. It was discovered by Hendrik Antoon Lorentz a Dutch physicist. It states that a particle of charge e moving through the presence of an electric field \mathbf{E} and magnetic field \mathbf{B} with velocity \mathbf{u} will suffer a force that will include the electrostatic force $e\mathbf{E}$ plus a term equal to $e\mathbf{u} \times \mathbf{B}$.

The collisional terms must be explained. The collisions that occur in a plasma are different from the strong and inelastic collisions that uncharged particles suffer because of the presence of Coulomb forces exerted by surrounding particles. As the collisions present are so complex, it is necessary to simplify these terms as means of collisional frequencies and rates, in order to treat charged particle collisions in a similar manner to uncharged particles. Lets assume that terms \mathbf{P}_{ei} and \mathbf{P}_{ie} describe the way both species exchange the momentum and are not dissipative. Therefore the two-fluid model states that $\mathbf{P}_{ei} = -\mathbf{P}_{ie}$, so that these terms cancel each other in the total momentum equation for the plasma. For that purpose, adding both momentum equations:

$$m_i n \frac{D\mathbf{u}_i}{Dt} = ne (\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \nabla p_i - ne (\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \nabla p_e \quad (3.7)$$

Knowing that $\nabla p = \nabla p_e + \nabla p_i$ and that $\rho = m_i n$, finally the total momentum equation for the plasma is formulated:

$$\rho \frac{D\mathbf{u}_i}{Dt} = ne (\mathbf{u}_i - \mathbf{u}_e) \times \mathbf{B} - \nabla p \quad (3.8)$$

that turns into

$$\rho \frac{D\mathbf{u}_i}{Dt} = \mathbf{j} \times \mathbf{B} - \nabla p \quad (3.9)$$

by using the general definition for the current produced by a two fluid plasma:

$$\mathbf{j} = ne (\mathbf{u}_i - \mathbf{u}_e) \quad (3.10)$$

Although the electric field disappears from the total momentum equation for the plasma in an quasineutral plasma, it is the main originator of the acceleration process. Paying attention to each of the species separately, it can be seen that the main driving force that increases the momentum of ions is in fact the collisions with electrons. In contrary the electrons are accelerated by the slip of velocities between them and the ions, provoked by the presence of the radial electric field, resulting in the global current.

Returning to collisions it is important to define the conductivity concept σ that is directly related to \mathbf{P}_{ei} and \mathbf{P}_{ie} by

$$\mathbf{P}_{ie} = -\mathbf{P}_{ei} = -\nu_{ei} m_e n (\mathbf{u}_i - \mathbf{u}_e) = -\frac{ne}{\sigma} \mathbf{j} \quad (3.11)$$

The scalar conductivity is therefore defined as:

$$\sigma = \frac{ne^2}{\nu_{ie} m_e} \quad (3.12)$$

Using (3.12), (3.4) and (3.5) can be expressed as

$$m_i n \frac{D\mathbf{u}_i}{Dt} = ne (\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \nabla p_i - \frac{ne}{\sigma} \mathbf{j} \quad (3.13)$$

$$0 = -ne(\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \nabla p_e + \frac{ne}{\sigma} \mathbf{j} \quad (3.14)$$

From (3.14), the so-called generalized Ohm's law can be postulated solving for \mathbf{j}

$$\mathbf{j} = \sigma \left(\mathbf{E} + \mathbf{u}_e \times \mathbf{B} + \frac{\nabla p_e}{ne} \right) = \sigma \left(\mathbf{E} + \mathbf{u}_i \times \mathbf{B} + \frac{\nabla p_e}{ne} - \frac{\mathbf{j} \times \mathbf{B}}{ne} \right) \quad (3.15)$$

Let us solve for the electric field in (3.15) to identify the different contributions.

$$\mathbf{E} = \frac{\mathbf{j}}{\sigma} - \frac{\nabla p_e}{ne} + \frac{\mathbf{j} \times \mathbf{B}}{ne} - \mathbf{u}_i \times \mathbf{B} \quad (3.16)$$

- $\frac{\mathbf{j}}{\sigma}$ is the Ohmic contribution.
- $-\frac{\nabla p_e}{ne}$ is the equivalent field of pressure gradient.
- $\frac{\mathbf{j} \times \mathbf{B}}{ne}$ field associated with the Hall current.
- $-\mathbf{u}_i \times \mathbf{B}$ is the electromotive force created by the ion motion across the magnetic field.

3.1.3 Performance: Mechanical work

Another step in the understanding process of the physics of these engines is the estimation of qualitative conclusions concerning their performance. Applying the scalar product of (3.9) and (3.10) with \mathbf{u}_i and \mathbf{u}_e respectively, we obtain:

KINETIC ENERGY EQUATION FOR ION SPECIES

$$\frac{D}{Dt} \left(\frac{1}{2} \rho u_i^2 \right) = -\nabla p_i \cdot \mathbf{u}_i + (\mathbf{j} \times \mathbf{B}) \cdot \mathbf{u}_i \quad (3.17)$$

The increment in kinetic energy of the flow is produced by two contributions.

- Main contribution: the work produced by the Lorentz force.
- Minor contribution: the work produced by the pressure gradient.

Now, let us look at the results obtained from the electron species. Applying the scalar product of (3.16) with \mathbf{j} :

$$\mathbf{j} \cdot \mathbf{E} = \mathbf{j} \cdot \frac{\mathbf{j}}{\sigma} - \mathbf{j} \cdot \frac{\nabla p_e}{ne} + \mathbf{j} \cdot \frac{\mathbf{j} \times \mathbf{B}}{ne} - \mathbf{j} \cdot \mathbf{u}_i \times \mathbf{B} \quad (3.18)$$

This latter equation can be simplified to

$$\mathbf{j} \cdot \left(\mathbf{E} + \frac{\nabla p_e}{ne} \right) = \frac{j^2}{\sigma} + \mathbf{u}_i \cdot (\mathbf{j} \times \mathbf{B}) \quad (3.19)$$

where

- $\mathbf{j} \cdot \left(\mathbf{E} + \frac{\nabla p_e}{ne} \right)$: Electric and pressure work
- $\frac{j^2}{\sigma}$: Ohmic heating
- $\mathbf{u}_i \cdot (\mathbf{j} \times \mathbf{B})$: Mechanical work applied electromagnetically.

The second term represents a *sink* of momentum in the mechanical energy balance. This term $\frac{j^2}{\sigma}$ express the loss of momentum or energy that appears from the conversion of ordered motion of electrons, into random electron motion. Commonly called *Joule* or *Ohmic Heating*, this term is an irreversible loss of energy as heat.

This analysis allows to reach some interesting conclusions. The acceleration mechanism based on the Lorentz force is dissipative. The energy loss occurs due to the collisional energy transfer between ions and electrons. Therefore, this technology entails an intrinsic inefficiency on its acceleration mechanism, compared to other EP systems. For further conclusions concerning the efficiency, the total energy equation should be formulated.

For $|\nabla P| \ll |\mathbf{j} \times \mathbf{B}|, |en\mathbf{E}|$ the whole picture becomes clearer.

3.2 Ideal SF-MPDT

As explained in (3.9) the total momentum gain of ions in MPD plasma is obtained by electrothermal (pressure force) and electromagnetic means (Lorentz force). For highly efficient MPD thrusters, the operation regime must be that for the electromagnetic thrust dominance. For that purpose, the thrust generated only by this contribution is studied in the following analysis.

The electromagnetic force \mathbf{F}_{EM} induced by an azimuthal magnetic field $\mathbf{B} = B_\theta \mathbf{i}_\theta$ can be calculated by integrating in the chamber volume the Lorentz Force.

$$\mathbf{F}_{EM} = \iiint_V \mathbf{j} \times \mathbf{B} dV = \frac{1}{\mu_0} \iiint_V (\nabla \times \mathbf{B}) \times \mathbf{B} dV \quad (3.20)$$

Then using the vector identity

$$(\nabla \times \mathbf{B}) \times \mathbf{B} = \mathbf{B} \nabla \cdot \mathbf{B} - \nabla \left(\frac{B^2}{2} \right) \quad (3.21)$$

the final expression for \mathbf{F}_{EM} is reached.

$$\mathbf{F}_{EM} = \frac{1}{\mu_0} \iiint_V \mathbf{B} \nabla \cdot \mathbf{B} dV - \frac{1}{\mu_0} \iiint_V \nabla \left(\frac{B^2}{2} \right) dV \quad (3.22)$$

$$= -\frac{1}{\mu_0} \iiint_V \left(\frac{B_\theta^2}{r} \mathbf{i}_r + \nabla \frac{B_\theta^2}{2} \right) dV \quad (3.23)$$

Notice that the resulting force has an axial and a radial component. The radial component is a compressive force defined by the following formula:

$$F_r = -\frac{1}{\mu_0} \iiint_V \left(\frac{B_\theta^2}{r} + \frac{\partial}{\partial r} \frac{B_\theta^2}{2} \right) dV = -\frac{1}{\mu_0} \iiint_V \frac{1}{r^2} \frac{\partial}{\partial r} \frac{r^2 B_\theta^2}{2} dV \quad (3.24)$$

and an axial force defined by:

$$F_z = -\frac{1}{\mu_0} \iiint_V \frac{\partial}{\partial z} \frac{B_\theta^2}{2} dz dA \quad (3.25)$$

Assuming constant area $A_z = \pi (r_a^2 - r_c^2)$ the axial forces is expressed by the following equation.

$$F_z = \frac{1}{2\mu_0} \int_{r_c}^{r_a} B_\theta^2 2\pi r dr \quad (3.26)$$

Finally, by using (3.1), that in this case states that $B_\theta = \frac{\mu_0 I_d}{2\pi r}$ being I_d the intensity of the current delivered,

$$F_z = \frac{\pi}{\mu_0} \int_{r_c}^{r_a} \left(\frac{\mu_0 I_d}{2\pi r} \right)^2 r dr = \frac{\mu_0 I_d^2}{4\pi} \ln \left(\frac{r_a}{r_c} \right) \quad (3.27)$$

This last equation was firstly formulated by Heinz Maecker, a German physicist, in 1955 [16].

Two different interesting facts can be deduced from this formula. Firstly, the axial force or thrust provided by the engine is independent on the size of the engine. Secondly, the thrust is proportional to the square of the current intensity.

Further corrections have been made, as parameter A that accounts for a more complex channel geometries [5].

$$F_z = \frac{\pi}{\mu_0} \int_{r_c}^{r_a} \left(\frac{\mu_0 I_d}{2\pi r} \right)^2 r dr = \frac{\mu_0 I_d^2}{4\pi} \ln \left(\frac{r_a}{r_c} + A \right) \quad (3.28)$$

Tested complex configurations for the channels change the relationship, following a law of the character $F_z = a I_d^2$. Introducing a characteristic parameter called $k = \frac{I_d^2}{\dot{m}}$

that as it is explained later, plays an important role in the appearance of the *onset phenomena*, the effective exhaust speed u_E can be written as:

$$u_E = \frac{F_z}{\dot{m}} = a \frac{I_d^2}{\dot{m}} = ak \quad (3.29)$$

The ideal kinetic power applied to the fluid is expressed as $P_T = \frac{1}{2} \dot{m} u_E^2 = \frac{a^2 k}{2} I_d^2$.

The total power required is just $P_{req} = \frac{P_T}{\eta_t}$. Therefore, expressions for the power required and the voltage can be easily inferred:

$$P_{req} = \frac{a^2}{2\eta_t \dot{m}} I_d^4 \quad (3.30)$$

$$V_{req} = \frac{a^2}{2\eta_t \dot{m}} I_d^3 \quad (3.31)$$

Concluding, the ideal SF-MPDT operational parameters have a direct relationship with the electric current, following:

$$F_z \propto I_d^2 \quad V_{req} \propto I_d^3 \quad P_{req} \propto I_d^4$$

3.3 MPDT Main Parameters

MPD thrusters accelerate the plasma by both thermal pressure gradient and the Lorentz force. The first MPD parameter compares both and is called the *plasma parameter* β . This parameter is equal to the ratio between the thermal pressure and the magnetic pressure.

$$\beta = \frac{2\mu_0 p}{B^2} = \frac{c_s^2}{c_A^2} = \frac{\text{thermal pressure}}{\text{magnetic pressure}} \quad (3.32)$$

where c_s is the sonic speed and c_A is the Alfven speed, defined as:

$$c_s = \sqrt{\frac{T_e}{m_i}} \quad \text{and} \quad c_A = \frac{B}{\sqrt{2\mu_0 m_i n}}$$

At low currents (powers) the magnetic pressure is small and the MPD thruster operates as an electrothermal device. In this regime MPDT has a poor performances because of ionization and electrode losses. However, when increasing the current, the magnetic field induced increases as well and the plasma parameter decreases. When the MPDT operates at $\beta < 1$, it is much more efficient. In fact, the behavior changes the dependance of the voltage from $V_d \propto I_d$ to $V_d \propto I_d^3$.

Moreover, there exist two other important MPDT parameters. They are the *Hall parameter* χ and the *Magnetic Reynolds Number* Rm . These parameters are defined by:

$$\chi = \frac{\omega_{ce}}{\nu_e} = \frac{\text{cyclotron frequency}}{\text{collisional frequency}} \quad \text{and} \quad Rm = \sigma u_i \mu_0 L = \frac{\text{emf}}{\text{resistive force}}$$

When the engine operates at low β the Rm uses to be large as $Rm \propto \beta^{-1}$. As it will be seen in the acceleration model, when the magnetic Reynolds number is large the electromotive force produced by the axial accelerated plasma cancels most of the radial electric field. Therefore, the currents in this region are almost inexistent and it concentrates in two regions one at the entrance and one at the exit. The main reason of this process is that the electromotive force is cancelled either due to almost null speed at the entrance or due to null magnetic field at the exit, and as a result, currents appear.

The Hall parameter is typically of the order unity for MPDT electromagnetic operation. If it is small, the current is mostly radial. As it is increased, the result is a growing axial contribution to the current vector. The hall parameter is related to the confinement of the plasma. It is defined as the ratio between two frequencies. The colloidal frequency increases with the intensity of the magnetic field and the collisional frequency increases with the plasma density. Therefore poor confinement conditions occur at low magnetic fields in dense and cold plasmas. The hall parameter for a MPDT uses to be $\mathcal{O}(1)$.

Therefore, the typical conditions on an MPDT are $\beta \ll 1$, $Rm \gg 1$ and $\chi \approx 1$.

3.4 Onset phenomena

A main limitation in the *steady-state* operation of an MPDT is the so called *onset phenomena*. This set of disturbing phenomena limits the power to be below a maximum value. Above this value, the operational behavior of the MPD worsens. Between the processes comprised in the so called onset, arise strong voltage fluctuations and increased electrode erosion. A large amount of experimental data has proven this phenomena. The limitation has been discovered experimentally to be based on a critical value of the parameter k [5][4].

Several theories try to assess an explanation to this phenomena. The two main theories are anode starvation and full ionization. The anode starvation theory states that the onset phenomena arises when the current density at the anode reaches the thermal current density and then the sheath structure in the anode disappears, creating detachment of the plasma from the anode and then increasing the possibility of potential fluctuations and vacuum arcs. On the other hand, the full ionization theory states that once the plasma is fully ionized, the energy transfered by arcs heats the electrode walls, leading to onset phenomena

None of this theories has been proved and this phenomena is still one key issue in the performance and operation of real MPD thrusters.

3.5 Prototypes

The history of the MPDT prototyping is really complicated since this technology required so much energetic effort to be tested that nowadays is still inoperative in space and the economical costs of tests are enormous. The latter appears due to the necessity of really advanced pumping systems for appropriate vacuum chambers and the power required to test these engines continuously during hours.

Along the history of electric propulsion several MPDT prototypes have been manufactured and the results were interesting but not profitable. Intermitent efforts have been made for roughly 60 years. Several prototypes have been manufactured, and tested, typically in quasi-steady (QS) operation. In fact data obtained from the majority of tests driven on these thrusters are from QS operation as steady operation at 1MW level is difficult. In 1995, Japan tested the first and last operational MPD flown in space, as a part of the Electric Propulsion Experiment (EPEX)[22] . The MPD thruster operated in quasisteady mode. Within few days of the experiment, the MPD arcjet operated by continuous and repetitive firings (over 40,000). The thruster was proved to perform as predicted by ground tests.

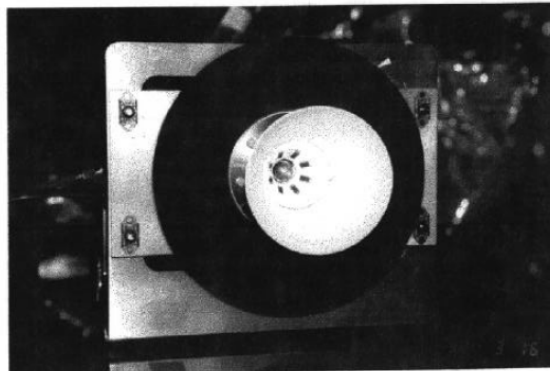


Figure 3.2: EPEX MPD arcjet thruster tested in space [22].

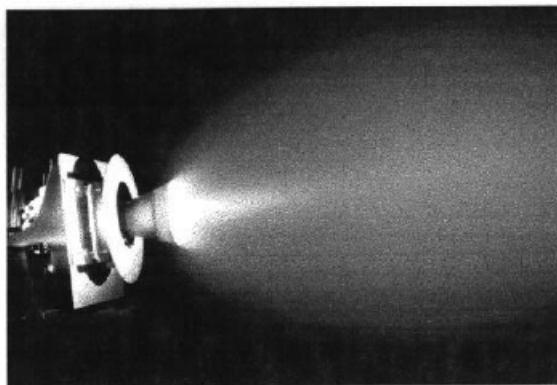


Figure 3.3: EPEX MPD arcjet thruster being tested in vacuum chamber[22].

This technology as explained before has the highest **thrust density** among the electric propulsion systems and a high specific impulse. As a direct consequence the most important research laboratories on EP, do not leave behind the research on this technology.

This section describes the main advanced prototypes that are dealing with the engines described in the models performed, the **self-field** magnetoplasmadynamic thrusters. However as it will be seen, the variation of the cross-section area with z is a constant design output for these thrusters and the models described do not include this option.

The principal MPD R&D loci are briefly mentioned hereinafter. This field shows a growing participation in research projects, with the development of models and thrusters to compare ideal and real operation.

The Princeton benchmark MPD Thruster

The Electric Propulsion Laboratory of Princeton University has developed a thruster that is known as the Princeton benchmark thruster. This thruster is a reference model of a self-field, coaxial, gas fed, and quasi-steady MPD.

The following sketch shows the geometry of this engine.

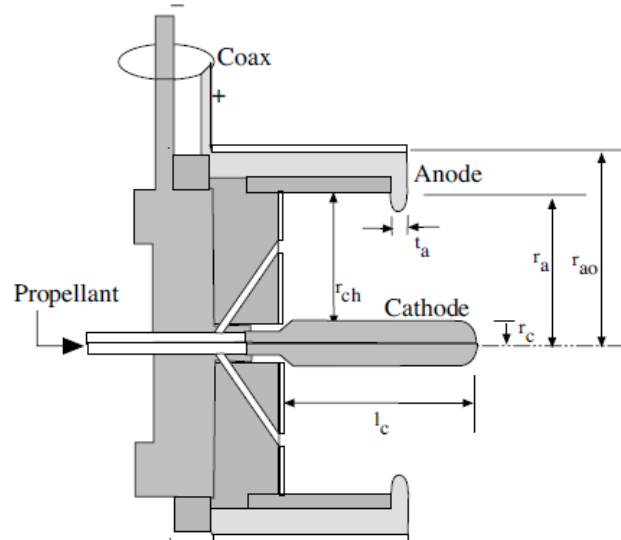


Figure 3.4: Schematic of the Princeton full-scale benchmark thruster. The dimensions are $r_c = 0.95$ cm, $r_a = 5.1$ cm, $r_{ao} = 9.3$ cm, $r_{ch} = 6.4$ cm, $t_a = 0.95$ cm and $l_c = 10$ cm.[10]

It consists of a 10 cm long hemispherically tipped, 2 cm diameter tungsten cathode in a 5.0 cm deep, 12.7 cm in diameter cylindrical chamber surrounded by an annular anode with a 1.0 cm thick lip of inner radius 5.1 cm.

In the interior is made of two coaxial parts. The cathode, which is made of thoriated tungsten, and the anode, which is an annular aluminum disk. At the beginning of the chamber, is the backplate, which is an insulator. It is made of boron nitride and the side insulator is a Pyrex tube. The exterior is insulated with a nylon cover.

Propellant is injected through a solenoid valve feeding a choked multiple orifice which splits the flow such that 54% of the mass flow rate goes through an annulus around the cathode base and 46% goes through a ring of 12 holes in the backplate located at a radius of 3.8 cm [10].

In steady state (SS) operation the injected propellant is ionized and accelerated downstream. The propulsive force is basically electromagnetic. Electrothermal acceleration is also present and is estimated to contribute 20 - 30% of the total thrust at low specific impulses (500 - 1000 sec). However, it becomes less significant with increasing the specific impulse.

IRS SF-MPD Prototypes

In Stuttgart, there is a R&D locus on MPD thrusters in the *Institut für Raumfahrtsysteme* (IRS). Several prototypes have been developed. The most important SF-MPD examples could be classified in two types or series:

- DT series: MPD thrusters with convergent divergent nozzles.
- ZT series: MPD thrusters with cylindrical nozzles and variable section cathodes.

DT2 and ZT3 are examples of these two series.

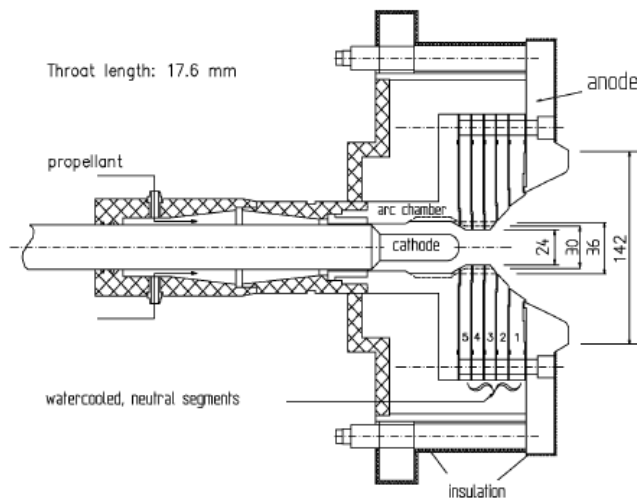


Figure 3.5: Schematic of the DT2 MPD thruster [6].

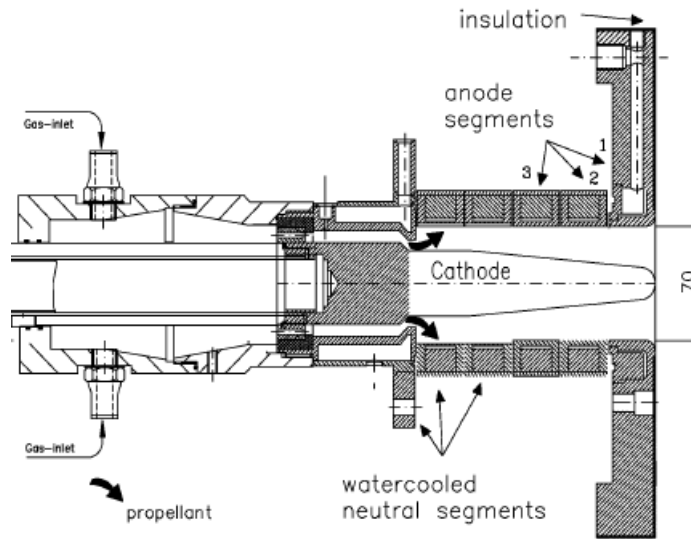


Figure 3.6: Schematic of the ZT3 MPD thruster [6].

The ZT3 is composed by three anode segments 2 neutral segments and a backplate. It has a copper cooling system with water and a thoriated tungsten cathode. The cathode has a radius that linearly decreases from 40 mm at the beginning to 10 mm at the tip.

As has been mentioned in *Real SF-MPD thrusters*, one of the main parameters that affect the operation of these thrusters is the value of k . It has been experimentally proven that there is a maximum of this parameter for which the operation becomes unstable and the *onset phenomena* appears. The tolerance of ZT3 has been proven to be higher than the one of DT6. Both tested at 2g/s of argon mass flow rate, while ZT3 onset did not appear till values of $k = 8 \times 10^{10} \text{ A}^2\text{s/kg}$, in the DT6 occurred at $k = 2.7 \times 10^{10} \text{ A}^2\text{s/kg}$. Notice that DT6 geometry is equivalent to that of DT2, but the throat diameter is 36 mm instead of the 24 mm of the DT2. The ZT3 has proven to provide 10 N thrust at 12.7 kW at 2 g/s.

Not only the advantages of the thrusters studied in Stuttgart are so relevant in term of thrust, but also, these engines have proven that the SF-MPDT operation and performance depends widely on the geometry of either the cathode, the anode or both.

URSS Prototypes

In the Union of Soviet Socialist Republics (URSS), extensive research was focused on MPD technology. In fact, there is a long list of tests, prototypes, and studies made. URSS scientists discovered that these engines had a strong limitation on their operation in SS mode for heating issues. The use of alkali metals (Lithium), as well as other improvements are example of the activity that took place in URSS concerning MPD thrusters. Important tests as 500 h continuous operation of the 0.5 MW Lithium MPDT were performed.

Organizations as *Keldysh Research Centre (KeRC)*, *NPD "Energiya"*, *EBD "Fakel"*, *MAI* took part in the R&D activities on MPD in the URSS [13]. Although the experience obtained is enhanceable, only the main SF-MPD device developed is described.

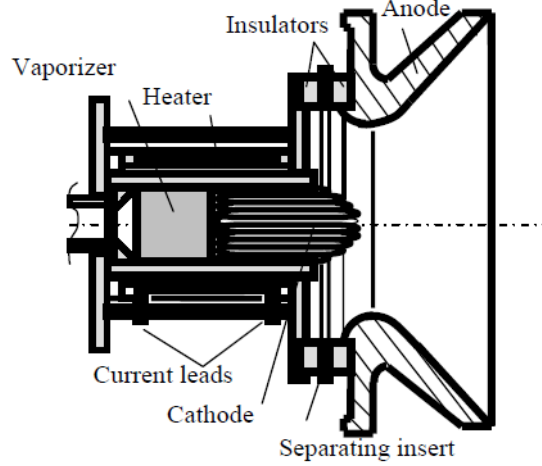


Figure 3.7: Schematic of KeRC MPD thruster [13].

The scheme of a 500 kW Lithium SFMPD thruster is presented in Fig.3.7. The use of a complicated cathode design allowed the operation at high current density when the cathode temperature was around 3000°C. The cathode was a multi-rod type (also known as multi-channel hollow cathode) with propellant passing through gaps between the rods. Usage of a Tungsten anode with radiation cooling eliminated the need of forced cathode cooling. The anode had a convergent-divergent shape.

This MPD, operating at 500 kW, demonstrated thrust of a several tens of Newtons level at exit flow velocities up to 80 km/s and efficiency up to 50%.

3.6 Applied-Field MPDT

The necessity to decrease the efficient power level MPD thrusters introduced the so-called AF-MPDT. These thrusters are potentially able to provide I_{sp} of the order of 5000 s, η_t greater than 50% at power levels between 10s and 100s of kW. AF low power (≤ 50 kW) MPD thrusters can provide similar performances to those of SF-MPDT, but decreasing considerably the thrust capability.

Extensive research is focus on these engines, whose phenomenology is complicated and three dimensional. Research activities started in the 1960s.

Typically, AF-MPD thrusters are coaxial accelerators that add an external magnetic field B_A (mostly axial, but small radial component) to the azimuthally self-induced magnetic field B_θ . The typical condition for these engines is that $B_A \gg B_\theta$. Commonly

the magnetic field lines diverge from the axial direction as they move away from the chamber (the magnetic coils). This fact can be noticed in Fig.3.8. The applied field produces a swirling current j_θ , which is known as *Hall Current*. By adding the external magnetic field, the device is able to operate at lower powers where B_θ is not enough to provide acceleration means to the propellant.

Therefore the acceleration process is much more complicated for the AF-MPDT than for the SF-MPDT. These thrusters use several acceleration mechanisms. The *Self-field acceleration*, *Swirl acceleration*, *Hall acceleration* and the *Gas dynamic acceleration*. The *self-field* part is the normal MPD acceleration process based on the Lorentz force that appears as a result of the interaction of the azimuthally induced magnetic field and the radial current ($j_r B_\theta$). The swirl component is produced by the interaction of the applied current with the applied field (both components), resulting in a Lorentz force with two contributions more, $j_z B_r$ and $j_r B_z$. When the hall parameters is high enough (low mass flow rates and strong magnetic fields), new two contributions appear to the acceleration, $j_\theta B_r$ and $j_\theta B_z$, resulting from the applied field and hall current induced by the perpendicularly applied magnetic and electric fields (E_r and B_z). Finally the gas dynamic acceleration takes place, which accounts for the plasma expansion and joule heating of the plasma contained in a magnetic nozzle (created by B_A). [15]

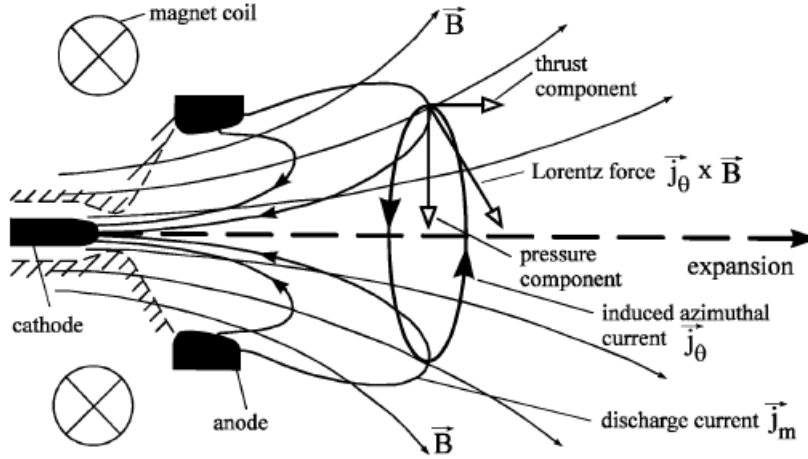


Figure 3.8: Schematic of an AF-MPDT [6].

3.7 Future challenges and research

Future research lines are based on solving the main important issues of these engines:

1. Electrode erosion
2. Plasma instabilities (*onset phenomena*)
3. Excessive power required, not only for space operation but also for ground testing

4. Power dissipated in the plasma by energy exchange process between electrons and ions

Several projects are trying to assess these issues with solutions that would increase the lifetime and performances. The use of complicated cathode geometries as the multi-hollow cathode or the use of liquid metal propellants as Barium or Lithium have increased the expectancies of these thrusters.

In fact a group of NASA researchers have developed the complete design of an advanced lithium-fed MPDT called α^2 . This thruster is a multihollow cathode lithium-fed AF-MPDT [12][5].

Another relevant project called CaLiPPSo was started and holds on at the moment. It involves the design and manufacture of a 500 kW SF-MPDT. The project is led by *Boeing* in collaboration with the *Jet Propulsion Laboratory*. An advanced cathode design, and the use of refractory metals that will radiatively cool, are some of the improvements this engine will incorporate. Both thrusters could potentially enable missions as fast robotic outer planet missions, Lunar and Mars cargo missions, piloted missions to Mars and even to outer planets beyond asteroid belt.

Chapter 4

Acceleration model

An complete axial model of the plasma discharge inside the MPDT must take into account both the processes of plasma production and plasma acceleration. This model will be analyzed in Chapter 5. From the experience and from that model, it is known that in certain operational range, ionization is limited to a thin region near the back plate. Downstream that region, both ionization and pressure effects become negligible, and the plasma is accelerated as a cold beam by the electromagnetic force. This situation is the one analyzed in this chapter. Once this model is understood and mastered we will afford the complete model, which turns out to be much more complex and stiff from points of view of numerical integration and parametric investigation.

4.1 Assumptions

In order to study the flow of a MPDT and to understand the basic parameters of the acceleration process of the flow in this type of engines, a model has been implemented with the following assumptions:

1. Plasma is already ionized.
2. Two species present: cations of type X^+ and electrons e^- .
3. Hall effects are negligible.
4. Pressure effects are negligible so plasma flow is considered hypersonic.
5. Cylindrical effects are neglected: quasiplanar thruster.
6. Steady $\left(\frac{\partial}{\partial t} = 0\right)$.
7. A single (averaged) electron temperature is used to define the collisional parameters of the model.
8. Quasiplanar assumption.
9. $\frac{dA}{dz} = 0$. Constant cross-sectional area.

4.2 Model description

The model requires as inputs the geometrical parameters of the engine anode radius r_a , cathode radius r_c and channel length L_z . Moreover it is needed to define the type of propellant (the mass m_i) and both parameters of operation: the current intensity I_d and the mass flow rate \dot{m} .

The main variables of the engine are:

- n : plasma density
- u : axial plasma (ion) velocity
- j : radial plasma current (ion+electrons)
- B : azimuthal induced magnetic field

4.2.1 Theoretical basis

The first step is to formulate the simplified plasma equations:

Mass Conservation

$$m_i n u = \Gamma_o \quad (4.1)$$

Momentum Conservation

$$m_i n u \frac{du}{dz} = j B \quad (4.2)$$

Ampere's Law

$$\frac{dB}{dz} = -\mu_0 j \quad (4.3)$$

Ohm's Law

$$j = \sigma(E - uB) \quad (4.4)$$

where $\sigma = \frac{e^2 n_e}{m_e \nu_e} = \frac{e^2}{m_e Q_{ei}}$.

Notice that σ is in fact σ_{11} of the conductivity tensor, as all other contributions result to be zero in one-dimensional flows. In fact the conductivity is the so-called parallel conductivity is inversely proportional to the collisional frequency. In this case is the ion-electron collisional frequency as no other particles are present in this model. Therefore, the conductivity of the plasma is inversely proportional to the number of

collisions that occur in an unitary volume. The formula defining this collisional frequency is typically the following for this model:

$$Q_{ei} = \frac{\nu_e}{n_e} = \frac{\nu_{ei}}{n_e} = 10^{-18} \left(\frac{1\text{eV}}{T_e} \right)^{3/2} \times \ln\Lambda \times 2.9 \times 10^6 \text{Hz} \times m^3 \quad (4.5)$$

Note that the electron temperature is given in electronvolts, therefore is not classical absolute temperature but an energetic temperature. The transformation from one to another only requires the use of Boltzmann constant $k_B \approx 1.38 \times 10^{-23} \text{J/K}$.

As can be noticed the conductivity is only a function of the temperature of the electrons T_e . Although $\ln\Lambda \sim 9 + \frac{1}{2} \ln \left[\left(\frac{10^{18} m^{-3}}{n_e} \right) \left(\frac{1\text{eV}}{T_e} \right)^3 \right]$ is a function of the temperature, its dependance is logarithmic and can be neglected for the sake of simplicity.

	$n_e(m^{-3})$	T(eV)	$\ln\Lambda$
Solar wind	10^7	10	26
Van Allen belts	10^9	10^2	26
Earth's ionosphere	10^{11}	10^{-1}	14
Solar corona	10^{13}	10^2	21
Gas discharge	10^{16}	10^0	12
Process plasma	10^{18}	10^2	15
Fusion experiment	10^{19}	10^3	17
Fusion reactor	10^{20}	10^4	18

Table 4.1: Typical values of $\ln\Lambda$

As the energy or temperature of the electrons in all plasmas for thrusters vary between 10^0 and few tens of eV, $\ln\Lambda$ will be constant. As a consequence, in our problem the conductivity will be roughly constant as the temperature will be assumed to be constant throughout the whole acceleration process.

Initially, the engine's geometry must be adapted to the quasiplanar approximation.

$$L_x = r_a - r_c \text{ and } L_y = \pi(r_a + r_c) \quad (4.6)$$

The magnetic field at the entry of the channel B_o can be calculated integrating (4.3) $I_d = \int_0^{L_z} j dz = \frac{B_o L_y}{\mu_0}$, leading to:

$$B_o = \frac{\mu_0 I_d}{L_y} \quad (4.7)$$

This equation can be compared to the magnetic field from a current in cylindrical coordinates which is $B_o = \frac{\mu_0 I_d}{2\pi r}$. In the cylindrical case the azimuthal magnetic field

induced increases the close the point is to the cathode ($r = r_c$). Integrating equation (4.2) using (4.3) a function of the axial velocity can be solved as:

$$u = \frac{B_o^2 - B^2}{2\mu_0\Gamma_o} = \left(1 - \frac{B^2}{B_o^2}\right) u_E \quad (4.8)$$

where, by imposing $B=0$ at $z=L_z$, u_E is obtained to be:

$$u_E = \frac{B_o^2}{2\mu_0\Gamma_o} = \frac{\mu_0}{2L_y^2\Gamma_o} I_d^2 \quad (4.9)$$

The latter equation states that the exit velocity and therefore the specific impulse is proportional to the current squared.

The thrust can be obtained by approximately $F \approx \Gamma_o u_E A$, which is:

$$F = p_{magnetic} A \equiv \frac{B_o^2 A}{2\mu_0} = \frac{\mu_0 I_d^2}{2} \frac{L_r}{L_y} \quad (4.10)$$

This result can be compared to the Maecker's law (3.27) for the thrust that is:

$$F_{cyl} = \frac{\mu_0 I_d^2}{4} \ln \frac{r_a}{r_c} \quad (4.11)$$

which allows for an estimation of the thrust of a cylindrical MPD thruster.

Concluding with the equations defining the model, (4.4) must be solved. Ohm's law can be expressed also as:

$$\frac{dB}{dz} = -\mu_0 j = -\mu_0 \sigma \left[E - \left(\frac{B_o^2 - B^2}{2\pi\Gamma_o} \right) B \right] \quad (4.12)$$

Finally some important thruster performance parameters can be calculated as the power used by the jet P_{jet} and a value for the propulsive efficiency (only taking into account the losses due to resistive medium) η_p , using their definitions.

The power induced to the flow during the acceleration process is the integral of the current density times the electromotive force over the total chamber volume:

$$P_{use} = A \int_0^{L_z} j u B dz = \frac{A}{\mu_0} \int_0^{B_o} B u dB = \frac{A u_E B_o^2}{\mu_0} \int_0^1 b(1 - b^2) db = \frac{F u_E}{2} \quad (4.13)$$

Then, the efficiency can be calculated directly from the ratio between the electric power consumed and the power induced to the flow.

$$\eta_p = \frac{P_{use}}{P_{QN}} = \frac{\frac{1}{2} \frac{\mu_0 I_d^2 L_r}{2 L_y} u_E}{\frac{I_d E L_r}{4 E}} = \frac{u_E B_o}{4 E} \quad (4.14)$$

4.2.2 Dimensional analysis

In most of physical problems it is useful to use dimensional analysis to nondimensionalize the variables and to find the minimum number of variables that define the behavior of the problem to be solved.

The model previously described, is found to have two non-dimensional parameters that define the behavior of the process. These nondimensional parameters are found to be:

- The magnetic reynolds number: $Rm \equiv \mu_o \sigma u_E L$
- The non-dimensional electric field: $\bar{E} = \frac{E}{u_E B_o}$

Applying dimensional analysis, one can notice that (4.2.1) nondimensionalized with the variables of the problem, can be expressed as:

$$\frac{db}{d\bar{z}} = -\mu_o \sigma u_E L_z (\bar{E} - (1 - b^2)b) = -Rm (\bar{E} - (1 - b^2)b) \quad (4.15)$$

where $b = \frac{B}{B_o}$ and $\bar{z} = \frac{z}{L_z}$.

The magnetic Reynolds is related to the conductivity or capability of charge transfer in the engine. Basically is proportional to the specific impulse of the thruster. The second parameter mentioned is the nondimensional electric field that is the ratio between the maximum reference magnetic component of the Lorentz force and the electric field. It is called reference force since it will never achieve that value because it is based on the magnetic field at the entrance of the chamber B_o and the speed at the exit u_E .

Notice that (4.15) is an ordinary differential equation on \bar{z} with a boundary condition and two nondimensional parameters. Actually the equation can be seen as a problem of a single parameter and an eigenvalue that must satisfy the boundary condition imposed.

Therefore the problem reduces to the solution of an ordinary differential equation that depends on two parameters, the magnetic Reynolds number and the non-dimensional electric field.

4.2.3 Matlab code

In order to solve (4.15) the code implements an iteration of the electric field to satisfy the boundary condition ($b = 0$ when $z = L_z$ or equivalently $\bar{z} = 1$).

The code calculates all parameters described $B_o, L_x, L_y, u_E, F, \sigma$, and therefore Rm for that conditions, as they are fixed once the dimensions L_z, r_c, r_a , electrons temperature T_e , propellant type mass m_i , electric current I_d and mass flow rate \dot{m} are given. Then it uses a function developed that solves the iteration explained looking for minimizing an error and seeks for the value of \bar{E} that satisfies the problem. A bisection

method was used after other trials for the iteration process. The code returns the solution of the acceleration process, yielding also the engine performance parameters described before.

4.2.4 Notes on thruster performance

The model shows that as the electric field must be higher than the magnetic field component of the Lorentz force, the minimum of \bar{E} must be equal to the maximum of the term on its right in (4.15), $(1 - b^2)b$. The maximum of that value is found to be $\frac{2}{3^{3/2}} \approx 0.38$. This is the minimum nondimensional electric field possible for any value of the Rm . Therefore, \bar{E} decreases as $Rm \rightarrow \infty$, to a minimum which is $E_{min} = \frac{2}{3^{3/2}}$.

Using the definition of \bar{E} , (4.14) turns into a much simpler form that is:

$$\eta_p = \frac{1}{4\bar{E}} \quad (4.16)$$

(4.16) shows that η_p is maximum for the minimum non dimensional electric field \bar{E}_{min} leading to the inequality $\eta_p \leq \frac{3^{3/2}}{8} \approx 0.65$.

This limit implies that there exists a minimum inevitable losses of energy do to resistivity, even when conductivity tends to infinity. Those energy losses are $P_{resistivity} = P_{QN} - P_{use}$, the so-called ohmic losses.

4.3 Results: Spatial response.

The model was used to analyze the flow axially to study the acceleration process. The results show the spatial response of the variables of the engine to the inputs described hereinafter. Notice that the Princeton's benchmark engine was use for this case but any engine of these characteristics could have been used.

Engine length L_Z [m]	Anode radius r_a [m]	Cathode radius r_c [m]
0.1	0.05	0.01
L_y [m]	L_r [m]	Cross-sectional area $A = L_r L_y [m^2]$
0.1885	0.04	0.00754

Table 4.2: Princeton Benchmark Geometrical data.

Propellant	Current intensity I_d [A]	Mass flow rate \dot{m} [kg/s]	T_e [eV]
Argon	23000	0.006	5

Table 4.3: Operational parameters selected

The numerical results of the model estimate the following operational parameters of the engine:

B_0 [T]	$G_0[kg/m^2s]$	u_E [m/s]	I_{sp} [s]	F_{jet} [N]	P_{jet} [W]
0.1533	0.7958	11755	1199	70.53	414579
$u_E B_0$ [V/m]	E [-]	E [V/m]	V_d [V]	P_{QN} [W]	η_p [-]
1803	0.3964	714.53	28.58	657366	0.63

Table 4.4: Engine performance parameters

As expected the velocity at the exit is relatively high close to a value of 12 km/s. Therefore the specific impulse is 1200 s which is relatively low but can be explained because of the non-cylindrical type of model and the high mass flow rate used. Normally convergent divergent nozzles could increase the specific impulse obtained as well but this cannot be studied by this model. Typically, these engines have high I_{sp} values as already mentioned in previous chapters.

The thrust is around 70 N which is a incredibly high value, even for the one expected from this type of engines. The power used by the jet is 412 kW, in the level expected. The actual value of the consumed power which is $P_{req} = P_{QN} = V_d \times I_d = E \times L_r \times I_d \approx 657$ kW. Therefore the efficiency is small but is below the maximum efficiency $\eta_{max} = 0.65$. From table 4.5 it can be appreciated that $Rm \approx 20$, allowing the propulsive efficiency of the engine to be close to the maximum, as can be appreciated in table 4.4. However this value would be diminished considerably by the sheath losses and the ionization losses which are present in these engines, which are not considered in this model and can mean up to a 50% of the total energy displayed.

Also it is relevant to notice the particle densities the engine deals with, which are around 10^{21} particles/ m^3 . Plasma densities are much higher than those achieved in other types of engines as ion engines or hall thrusters, that allows to obtain considerably high thrusts maintaining the specific impulse close to that of the other types of electromagnetic accelerators. The plasma operating in this type of electromagnetic accelerators is colder than in other types and also is more collisional.

ν_e [Hz]	n_E [part./ m^3]	$\sigma_{ }$ [A/Vm ²]	R_m [-]	Q_{ei} [$m^{-3}s^{-1}$]
2.1×10^9	1.02×10^{21}	13665	20.18	4.12×10^{-12}

Table 4.5: Flow characteristic parameters

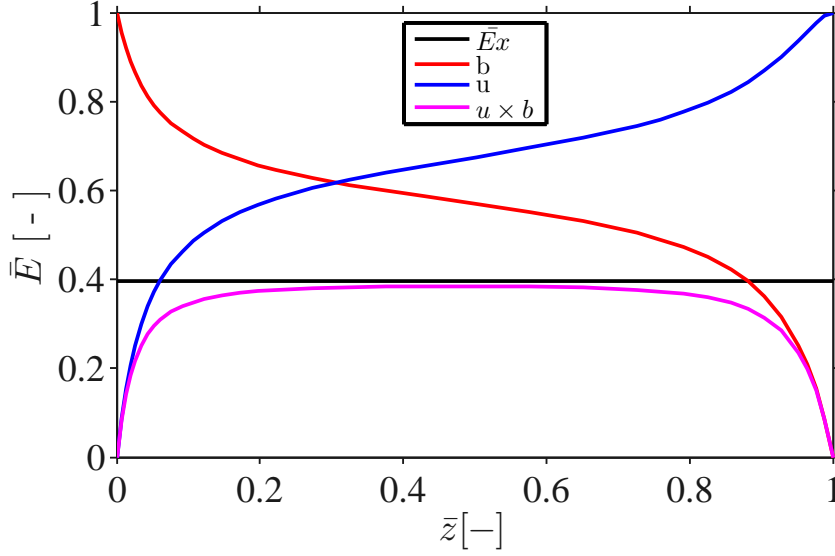


Figure 4.1: Evolution of the nondimensional variables along the longitudinal axis of the thruster

Furthermore the solution flow variables along the channel is obtained. The results are shown as nondimensional in figure 4.1. This result is for $\mathbf{Rm} \approx 20$. It can be appreciated that the induced electric field ($\mathbf{u} \times \mathbf{b}$) never exceeds the value of the minimum electric field.

The combination of both radial current and electric field, produces an axial electric field that generates an acceleration process that can be related with the other variables by:

$$E_z = \frac{\omega_{ce}}{\nu_{ei}} E_x = \frac{eB}{m_e Q_{ei} n_e} E_x \alpha \frac{B}{n_e} = \frac{eBu}{m_e Q_{ei} \Gamma_o} E_x \quad (4.17)$$

Therefore, $E_z \propto \frac{Bu}{\Gamma_o}$. If we nondimensionalize E_z as a function of E_x , the result shows

$$\frac{E_z}{u_E B_o} = \frac{u}{u_E} \frac{B}{B_o} \frac{E_x}{u_E B_o} \frac{eB_o u_E}{m_e Q_{ei} \Gamma_o} = \varepsilon \bar{E}_x \chi^* \quad (4.18)$$

This axial electric field E_z is shown therefore to be proportional to the electromotive force ε , the nondimensional electric field \bar{E} and a 'Hall Parameter' χ . The latter is another reference value that does not represent any realistic point as takes the entrance and exit for its quantities.

Another relevant fact is that increasing \mathbf{Rm} the layers of acceleration are decreased in thickness. The acceleration process occurs when current appears and the current in nondimensional form is $\bar{\mathbf{j}} = \bar{\mathbf{E}} - \mathbf{u} \times \mathbf{b}$. Therefore at higher magnetic reynolds numbers the flow accelerates only at the entrance and at the exit and abruptly instead of a gradual acceleration throught the channel obtained with low values of \mathbf{Rm} .

Basically as Rm increases the resistivity of the flow decreases and the currents produced are high but very concentrated, allowing a better acceleration process (higher efficiencies).

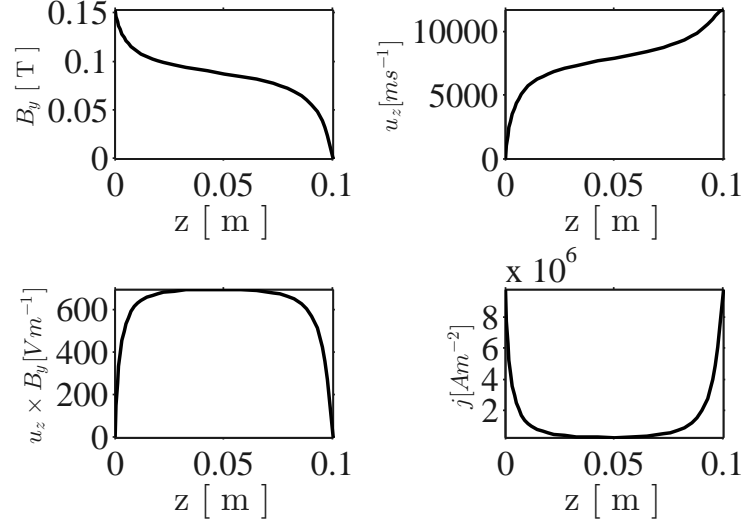


Figure 4.2: Spatial response of main variables

It can be noticed that gradually the magnetic field decreases transferring momentum to the flow while the induced electric field is maximum at the middle of the engine and minimum at the entrance and at the exit, allow for higher currents in those regions.

4.4 Results: Nondimensional performance maps

After nondimensionalizing the equations, it turns out that there is only a single parameter changing the character of the problem: the magnetic Reynolds number Rm . It is seen that for every single case, imposing the boundary condition of the magnetic field being B_o at the entrance of the channel and 0 at the exit, there is only a single solution for the electric field.

$\bar{E}(Rm)$

A study of this dependence was made to conclude what was the map of solutions of this type we could encounter. The following figure shows the results and afterwards the analysis of these results is made.

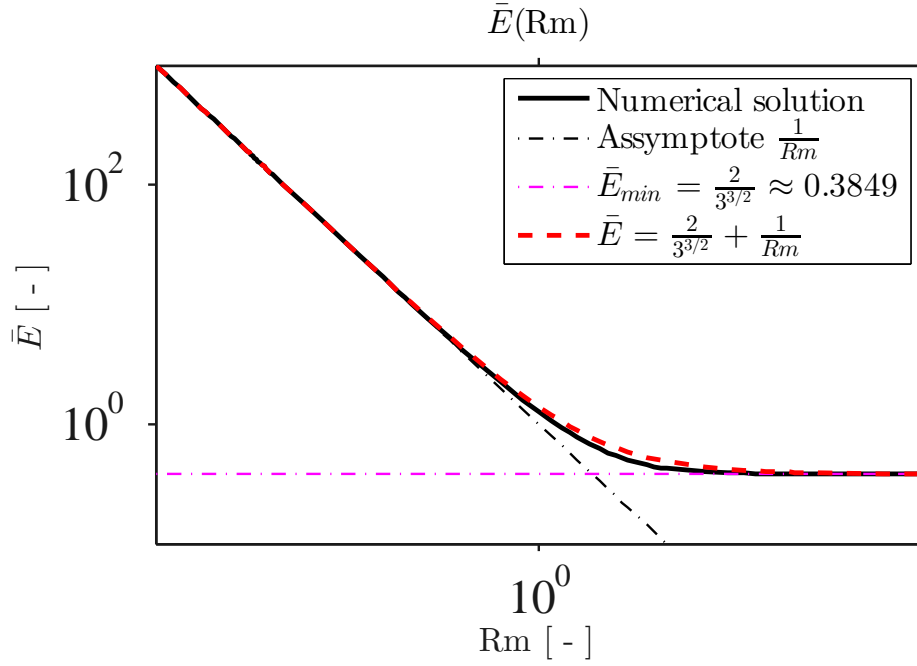


Figure 4.3: Map of the behaviour of the flow Electric field with the magnetic Reynolds number

The graph shows us two asymptotical behaviors; as $Rm \rightarrow 0$, $\bar{E} \rightarrow \infty$ and also that as $Rm \rightarrow \infty$, $\bar{E} \rightarrow \frac{2}{3^{3/2}}$. The function can be approximated by the following formula:

$$\bar{E} = \frac{2}{3^{3/2}} + \frac{1}{Rm} \quad (4.19)$$

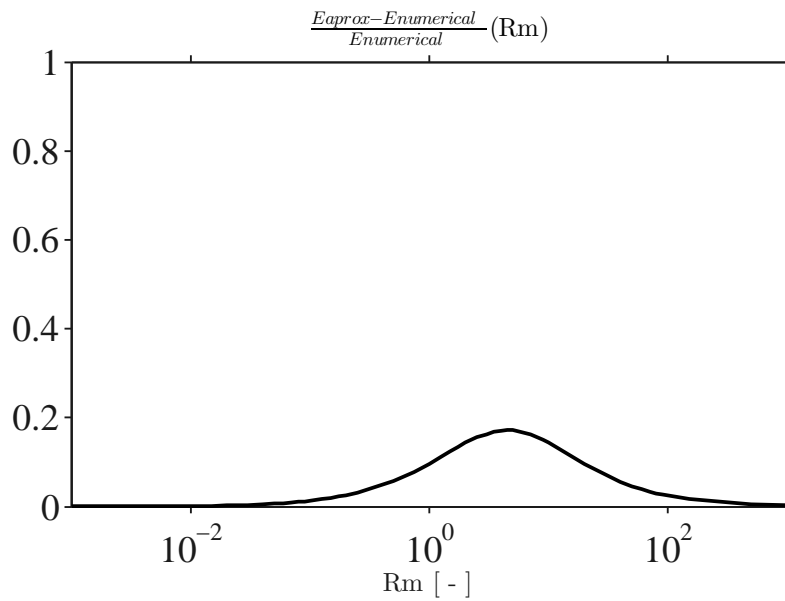


Figure 4.4: Error of the approximation formula

It can be appreciated that the error obtained from the comparison between the approximation stated in (4.19), and the accurate numerical solution, is the highest for values of Rm between 0.1 and 100. The maximum error is a 18% of \bar{E} . It could be considered that the error is small but in fact the thruster length showed high sensitivity to changes in the nondimensional electric field. Therefore, as the model computational time is considered to be small with the numerical solution, the exact solution is used as default by the code but a similar function using the approximated curve has been integrated as an option.

$\eta_p(Rm)$

The following figure has been obtained from the use of the propulsive efficiency definition and by relating it with the magnetic Reynolds number. The relation between \bar{E} and Rm obtained numerically has been combined with the theoretical law expressed in (4.16). This law was also combined with (4.19) to obtain the approximated law for the η_p as a function of Rm .

$$\eta_p = \frac{3^{3/2}}{4} \frac{Rm}{2Rm + 3^{3/2}} \quad (4.20)$$

The latter equation shows that for high enough magnetic Reynolds numbers, the efficiency reaches its maximum. This can be also be seen from the following figure.

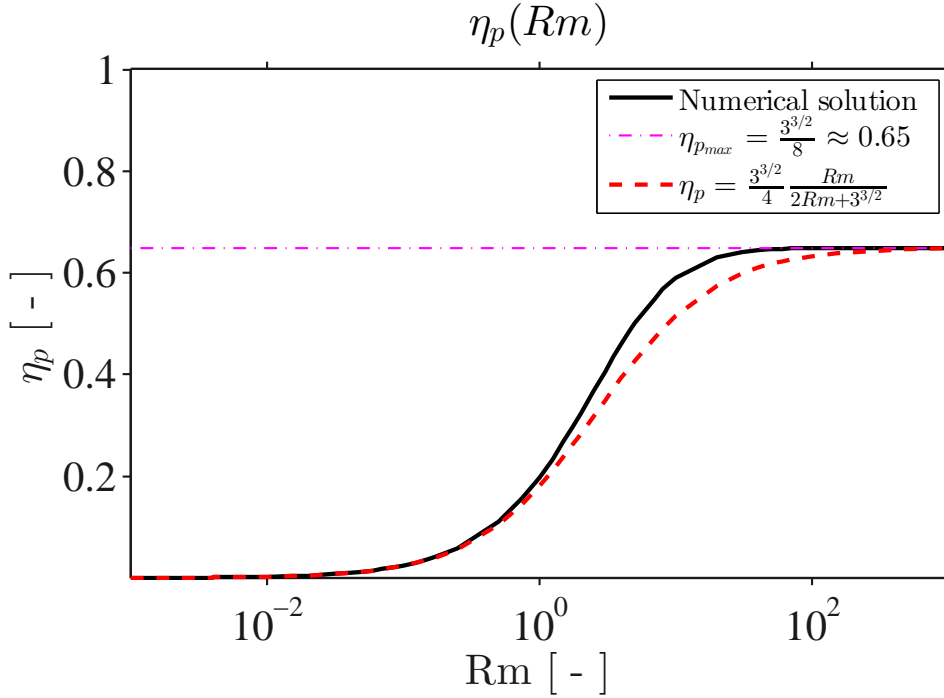


Figure 4.5: Efficiency map

In (4.19), when $Rm \approx 2.59$ both parts of the equation are equal so there is a

different behavior for $Rm \ll 2.59$ and for $Rm \gg 25.9$. The \bar{E} produced in the first case is very high as Rm is decreased. On the other hand as Rm is increased over this 2.59 the \bar{E} produced is very close to the value previously described. Therefore as was explained before it is recommendable to operate the engine at high values of the magnetic Reynolds number but an increase on Rm after obtaining $Rm \approx 50$, would not make a valuable difference in the performance of the engine.

4.5 Results: Parametric study

The model allows three main parameters to be varied, the mass flow rate \dot{m} , the current intensity I_d , and the electron temperature T_e . In order to analyze the flow, a study of the effect of these parameters on the thruster operation has been performed. Also the efficiency map is shown. Notice that this study is governed by the nondimensional curves shown in previous section, which represent any combination of dimensional parameters that result in that nondimensional case. In fact, this study has been carried out to improve the understanding of the real operation of these thrusters.

I_d : The current

Concerning the performance and cost, the parametric dependance w.r.t. the current intensity shows that these type of acceleration process requires higher voltages to operate at higher currents. Moreover, it can also be outlined that the higher the mass flow rate the higher the currents we can obtain with the same voltage. Also it must be enhanced that the efficiency rises as the current increases. Furthermore the lower the mass flow rate the higher the efficiency of the acceleration process. The latter can be explained by the increased collisionality of the plasma when densities are increased.

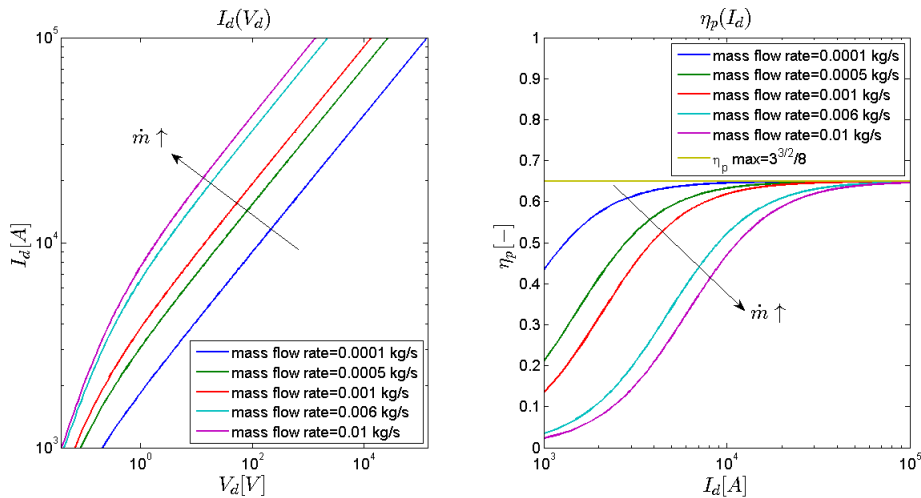


Figure 4.6: Parametric dependance w.r.t. current intensity

\dot{m} : mass flow rate

The parametric dependance w.r.t. the mass flow rate shows that the higher the flow rate the lower the voltage required.

As could be expected, the current we desire to obtain is proportional to the voltage required. The presence of low mass flow rates increases the efficiency of the process but also rises its cost. It is remarkable that there is a minimum voltage required to obtain the current as all $V_d(\dot{m})$ curves seem to tend asymptotically to that value when $\dot{m} \rightarrow \infty$. However at those points, the propulsive efficiency obtained from the acceleration process is incredibly small.

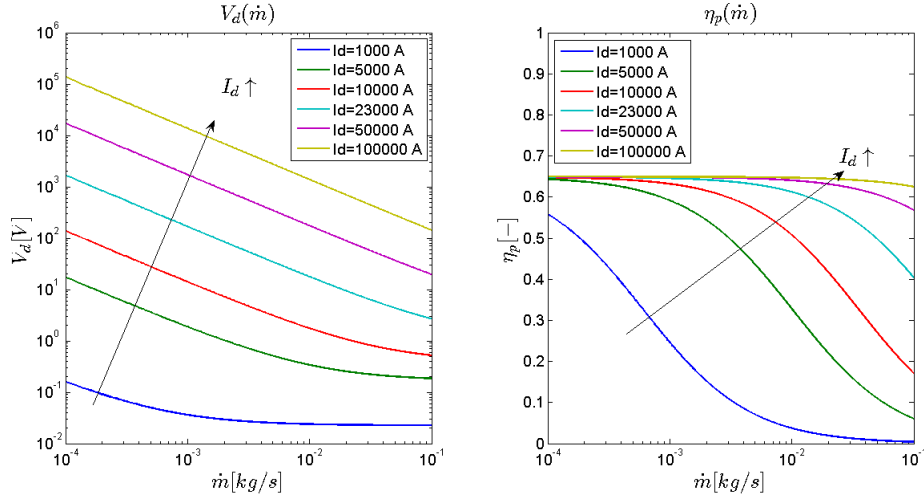


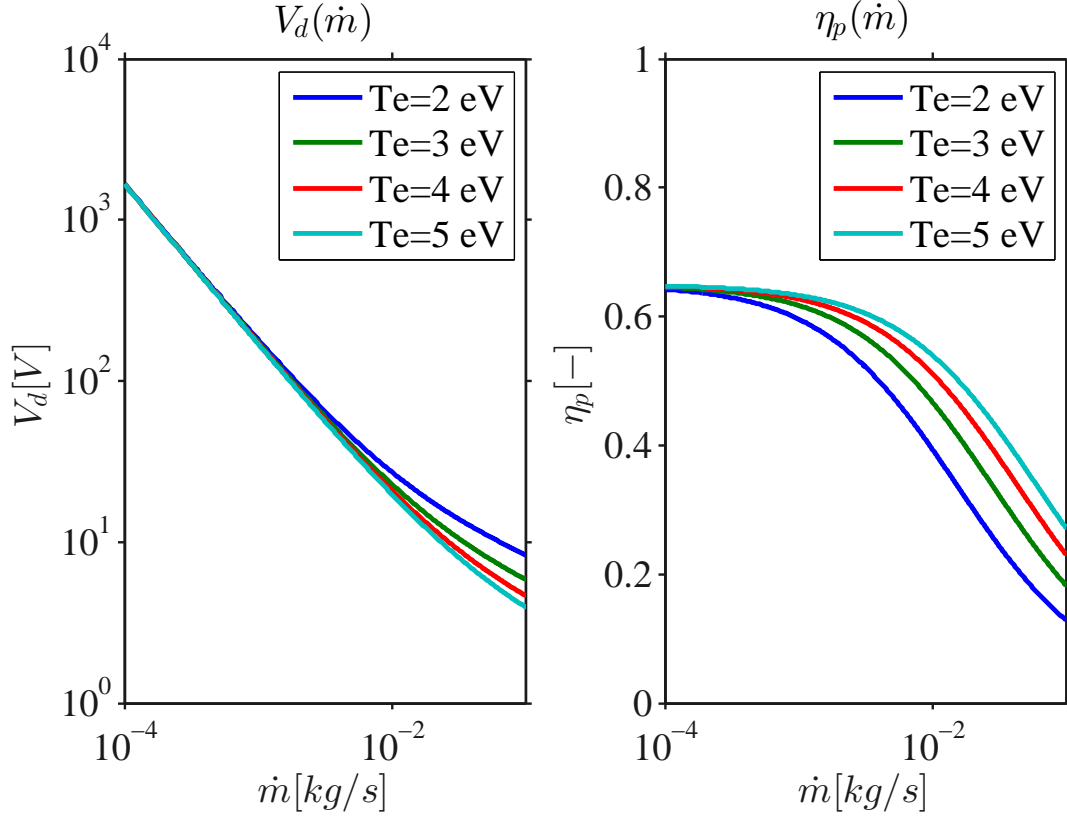
Figure 4.7: Parametric dependance w.r.t. mass flow rate

T_e : electron temperature

An analysis of the dependance of the electron temperature on the operation of the engine in this model has been carried out.

By increasing the temperature, the engine requires less voltage bias to provide a fixed a current at a fixed mass flow rate. Therefore increasing the temperature of the plasma has a positive effect on the cost.

In addition to that, the following figure also reveals that the efficiency of these thrusters is increased when the electron temperature is increased.

Figure 4.8: Parametric dependence w.r.t. T_e at 23000 A

Efficiency as a function of \dot{m} and P_d

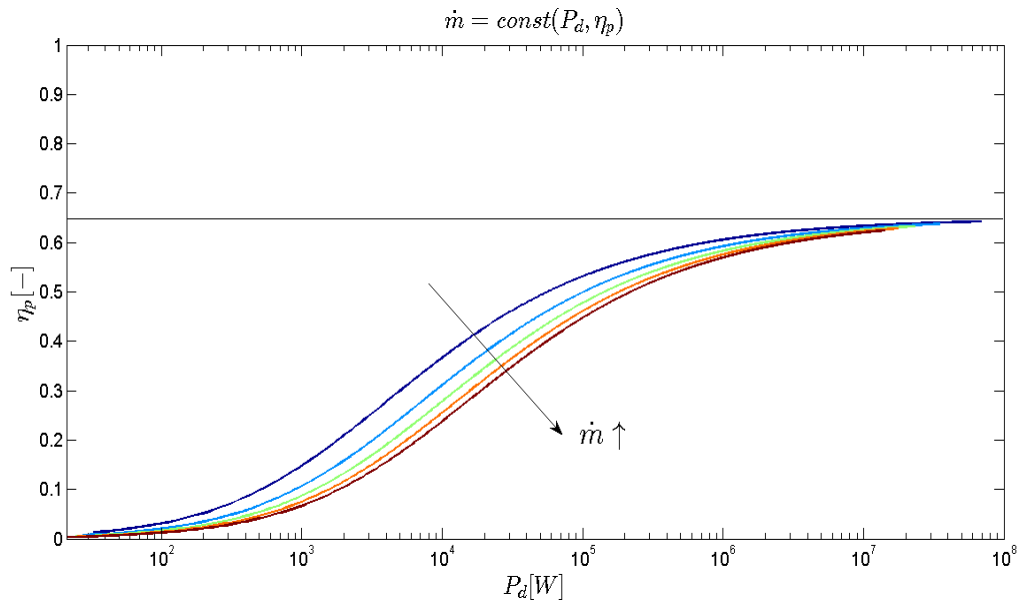


Figure 4.9: Efficiency as a function of the power consumption and the mass flow rate

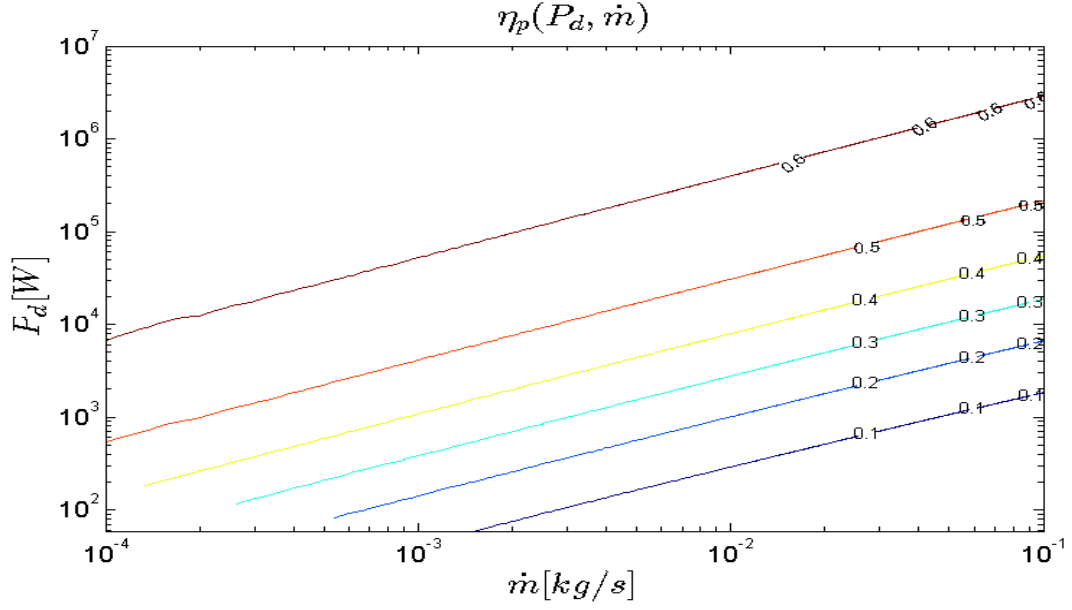


Figure 4.10: Power delivered as a function of the mass flow rate showing the lines of constant efficiency η_p .

The two latter figures represent the dependance of the propulsive efficiency with respect to the power and the mass flow rate. These two parameters are of main importance in propulsion systems since they are main limiters for the thruster viability.

It can be noticed that this model predicts that MPD thrusters operate efficiently in the hundreds of kilowatt range and even better in the megawatt range, a fact that experimental data has proved. Also the efficiency is seen to be constant for a proportional variation of both P_d and \dot{m} .

Concluding, the efficiency of the acceleration process is proportional to the current intensity but also increases the cost in voltage bias. However by increasing the mass flow rate the V_d can be lowered, lowering the efficiency of the acceleration process as well. As in normal engines the higher the amount of thrust the engine is desired to produce the lower the efficiency of the acceleration process. This series of facts requires a trade-off analysis where the optimum is dependent on the criteria used.

It must be also remarked that the parametric study undertaken using the acceleration model reveals that increasing the electron temperature has a positive effect on the operation of MPD thrusters.

Nevertheless this model shows that these engines are low cost in voltage bias and highly efficient at small mass flow rates but the performance just falls when \dot{m} is increased. Furthermore, by increasing the voltage difference between anode and cathode, a high performance with higher mass flow rates can be achieved.

Chapter 5

Complete Axial Model

In addition to the acceleration process produced by the Lorentz force, other processes occur in parallel and in fact all interact throughout the acceleration process. Typically plasma ionization can occur in a large region throughout the engine. As a consequence all collisional processes should be studied to seek for ionization scales. Moreover plasma pressure and therefore subsonic-supersonic transitions occur during the process, and are important mainly at low power. Both issues in addition to the acceleration model previously described in Chapter 4 are analyzed in this model.

The Complete Axial Model is represented in this chapter. Firstly, assumptions must be made in order to simplify the problem. Then, the chapter shows the model formulation and outlines the main important steps in the model development until the final mathematical problem is expressed. The final model is nondimensional. The boundary conditions are detailed several times throughout the chapter as these evolve with the model manipulation. Boundary conditions are imposed basically at the backplate, the sonic and the exit points. Finally, after the Matlab code is described, the results are shown and analyzed.

5.1 Assumptions

The main assumptions made on this model are:

- Plasma is produced via ionization of the neutral gas.
- Three species present: cations of type X^+ , electrons e^- and neutral atoms X .
- Hall effects are negligible.
- Pressure effects are not negligible.
- Sonic transition is studied.
- Cylindrical effects are neglected: quasiplanar thruster.
- Steady $\left(\frac{\partial}{\partial t} = 0\right)$.

- The flow is considered to be isothermal.
- Ion-electron, charge exchange, electron-neutral collisional processes and the ionization rate, are all taken into account.
- Electron inertia is neglected.
- Quasiplanar assumption.
- $\frac{dA}{dz} = 0$. Constant cross-sectional area.

5.2 Model development

5.2.1 1st set of equations

Initially, let us begin with the set of plasma vectorial equations in differential form.

The generation of ionized mass is called S_i and its a function of the ionization rate Q_{ion} and the product of the densities of electrons and neutral atoms.

$$\nabla \cdot n_i \mathbf{u}_i = S_i = n_e n_n Q_{ion}(T_e) \quad (5.1)$$

Mass Conservation

$$\nabla \cdot (n_n \mathbf{u}_n - n_i \mathbf{u}_i) = 0 \quad (5.2)$$

Current Conservation

$$\nabla \cdot \mathbf{j} = 0 \quad (5.3)$$

Electrostatic potential definition

As the electric field is irrotational ($\nabla \times \mathbf{E} = \mathbf{0}$) and $\nabla \cdot \mathbf{E} = \frac{e}{\epsilon_0} (n_i - n_e)$, then the definition of the electrostatic potential is obtained. $\mathbf{E} = -\nabla \Phi$. Therefore,

$$\nabla^2 \Phi = \frac{e}{\epsilon_0} (n_i - n_e) \quad (5.4)$$

Ampere's Law

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} \quad (5.5)$$

As explained before Ampere's Law states that the presence of a current, induces a magnetic field. In addition to that, another property of the magnetic field was described by Maxwell. The solenoidal character of the magnetic field

$$\nabla \cdot \mathbf{B} = 0$$

Momentum Conservation for ion species

$$\nabla \cdot m_i n_i \mathbf{u}_i \mathbf{u}_i = \nabla P_i + e n_i (\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \mathbf{R}_i \quad (5.6)$$

where $\mathbf{R}_i = -m_e n_e^2 Q_{ei} (\mathbf{u}_i - \mathbf{u}_e) - m_i n_e n_n Q_{cx} (\mathbf{u}_i - \mathbf{u}_n)$, is the momentum exchange of ions with other species.

Momentum Conservation for neutral species

$$\nabla \cdot m_i n_n \mathbf{u}_n \mathbf{u}_n = -\nabla P_n - \mathbf{R}_n = m_i n_n \mathbf{u}_n \cdot \nabla \mathbf{u}_n - m_i S_i \mathbf{u}_n \quad (5.7)$$

where $\mathbf{R}_n = -m_e n_e n_n Q_{en} (\mathbf{u}_n - \mathbf{u}_e) - m_i n_e n_n Q_{cx} (\mathbf{u}_n - \mathbf{u}_i)$, is the momentum exchange of neutrals with other species.

Momentum Conservation for electron species

$$0 = -\nabla P_e - e n_e (\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \mathbf{R}_e \quad (5.8)$$

where $\mathbf{R}_e = -m_e n_e^2 Q_{ei} (\mathbf{u}_e - \mathbf{u}_i) - m_e n_e n_n Q_{en} (\mathbf{u}_e - \mathbf{u}_n)$, is the momentum exchange of electrons with other species.

Using the definition of current density (3.10), and from electrostatic potential, (5.8) turns into

$$\mathbf{0} = -\nabla P_e - e n (-\nabla \Phi + \mathbf{u}_i \times \mathbf{B}) + \mathbf{j} \times \mathbf{B} - \mathbf{R}_e \quad (5.9)$$

These equations define the behavior of variables $n_e, n_i, n_n, \mathbf{u}_e, \mathbf{u}_i, \mathbf{u}_n$ or $\mathbf{j}B$ and Φ .

5.2.2 Simplification to a 1D model

In order to go from a vectorial model to a onedimensional set of equations first assume that all $\frac{\partial}{\partial y} = 0$. Notice that the model is performed in cartesian coordinates. This choice does not affect the model as z direction is the only one shared between both coordinate systems.

Mass and momentum conservation equations

The mass and momentum conservation for ions and neutrals are formulated.

Assuming the quasineutrality condition (3.3), all the equations for electron species dissapear.

The *mass conservation* principle is now expressed by two differential equations for ions and neutrals.

$$\frac{\partial}{\partial x}(nu_{xi}) + \frac{\partial}{\partial z}(nu_{zi}) = S_i \quad (5.10)$$

$$\frac{\partial}{\partial x}(n_n u_{xn}) + \frac{\partial}{\partial z}(n_n u_{zn}) = -S_i \quad (5.11)$$

$$(5.12)$$

Similarly happens to the *momentum conservation* principle for both species, which results in the following four equations.

IONS

$$m_i n \left(\frac{\partial u_{xi}}{\partial x} + \frac{\partial u_{zi}}{\partial z} \right) u_{xi} = -\frac{\partial P}{\partial x} + j_y B_z - j_z B_y - R_{xi} - S_i m_i u_{xi} \quad (5.13)$$

$$m_i n \left(\frac{\partial u_{xi}}{\partial z} + \frac{\partial u_{zi}}{\partial x} \right) u_{zi} = -\frac{\partial P}{\partial z} + j_x B_y - j_y B_x - R_{zi} - S_i m_i u_{zi} \quad (5.14)$$

$$(5.15)$$

NEUTRALS

$$m_i n_n \left(\frac{\partial u_{xn}}{\partial x} + \frac{\partial u_{zn}}{\partial z} \right) u_{xn} = -\frac{\partial P_n}{\partial x} - R_{xn} + S_i m_i u_{xn} \quad (5.16)$$

$$m_i n_n \left(\frac{\partial u_{xn}}{\partial z} + \frac{\partial u_{zn}}{\partial x} \right) u_{zn} = -\frac{\partial P_n}{\partial z} - R_{zn} + S_i m_i u_{zn} \quad (5.17)$$

$$(5.18)$$

Equations for \mathbf{j} , \mathbf{B} and Φ

Then, the equations for the \mathbf{j} , \mathbf{B} and Φ must be reformulated and simplified:

The current conservation principle turns into:

$$0 = \frac{\partial j_x}{\partial x} + \frac{\partial j_z}{\partial z} \quad (5.19)$$

From (5.9) the following equations are deduced

$$0 = -\frac{\partial P_e}{\partial x} + en \left(-\frac{\partial \Phi}{\partial x} + u_{yi} B_z - u_{zi} B_y \right) + j_y B_z - j_z B_y + \frac{m_e \nu_e}{e} j_x \quad (5.20)$$

$$0 = -en(u_{zi} B_x - u_{xi} B_z) + j_z B_x - j_x B_z + \frac{m_e \nu_e}{e} j_y \quad (5.21)$$

$$0 = -\frac{\partial P_e}{\partial z} + en \left(-\frac{\partial \Phi}{\partial z} + u_{xi} B_y - u_{yi} B_x \right) + j_x B_y - j_y B_x + \frac{m_e \nu_e}{e} j_z \quad (5.22)$$

Moreover, the following three differential equations appear from (5.5):

$$\mu_0 j_x = -\frac{\partial B_y}{\partial z} \quad (5.23)$$

$$\mu_0 j_y = \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \quad (5.24)$$

$$\mu_0 j_z = \frac{\partial B_y}{\partial x} \quad (5.25)$$

By imposing $j_y = 0$, $B_x = 0$, $B_z = 0$, and taking into account that the model is unidimensional (velocity is axial), four equations remain with (5.19) :

$$0 = -\frac{\partial P_e}{\partial x} - en\left(\frac{\partial \Phi}{\partial x} + u_{zi}B_y\right) - j_z B_y + \frac{m_e \nu_e}{e} j_x \quad (5.26)$$

$$0 = -\frac{\partial P_e}{\partial z} - en\left(\frac{\partial \Phi}{\partial z} - u_{xi}B_y\right) + j_x B_y + \frac{m_e \nu_e}{e} j_z \quad (5.27)$$

$$\mu_0 j_x = -\frac{\partial B_y}{\partial z} \quad (5.28)$$

$$\mu_0 j_z = \frac{\partial B_y}{\partial x} \quad (5.29)$$

Let us call

$$\mathbf{H} = -\nabla \Phi + \mathbf{u}_i \times \mathbf{B} + \frac{\nabla P_e}{en}$$

If we plug in this definition into the previous set defined we reach the two following equations directly:

$$0 = -enH_x - j_z B_y + \frac{m_e \nu_e}{e} j_x \quad (5.30)$$

$$0 = -enH_z + j_x B_y + \frac{m_e \nu_e}{e} j_z \quad (5.31)$$

Returning to the previous equations, introducing the definition of *gyrofrequency* $\omega_{ce} = \frac{eB}{m_e}$ and multiplying both equations by $\frac{e}{m_e}$:

$$-\omega_{ce} j_z + \nu_e j_x = \frac{e^2 n}{m_e} H_x \quad (5.32)$$

$$-\omega_{ce} j_x + \nu_e j_z = \frac{e^2 n}{m_e} H_z \quad (5.33)$$

Then, let us solve for j_x and j_z :

$$j_x = \frac{e^2 n \nu_e H_x + \omega_{ce} H_z}{m_e \nu_e^2 + \omega_{ce}^2} = \frac{e^2 n}{m_e \nu_e} \frac{H_x + \beta H_z}{1 + \beta^2} = \sigma_{\perp} H_x + \sigma_H H_z \quad (5.34)$$

$$j_z = \frac{e^2 n \nu_e H_z - \omega_{ce} H_x}{m_e \nu_e^2 + \omega_{ce}^2} = \frac{e^2 n}{m_e \nu_e} \frac{H_z - \beta H_x}{1 + \beta^2} = \sigma_{\perp} H_z - \sigma_H H_x \quad (5.35)$$

In these equations, a new parameter is introduced. It is the so-called *Hall Parameter* χ defined in previous chapters. Also two different conductivities are defined, the perpendicular transport σ_{\perp} and the hall transport σ_H (or conductivity). They are defined by:

$$\sigma_{\perp} = \frac{e^2 n}{m_e \nu_e} \frac{\nu_e^2}{\nu_e^2 + \omega_{ce}^2} = \sigma_{\parallel} \frac{1}{1 + \chi^2} \quad \text{and} \quad \sigma_H = \frac{e^2 n}{m_e \nu_e} \frac{\nu_e \omega_{ce}}{\nu_e^2 + \omega_{ce}^2} = \sigma_{\parallel} \frac{\chi}{1 + \chi^2}$$

Therefore, the set of equations for j_x, j_z, B_y are:

$$j_x = -\frac{1}{\mu_0} \frac{\partial B_y}{\partial z} = \sigma \frac{H_x + \chi H_z}{1 + \chi^2} \quad (5.36)$$

$$j_z = \frac{1}{\mu_0} \frac{\partial B_y}{\partial x} = \sigma \frac{H_z - \chi H_x}{1 + \chi^2} \quad (5.37)$$

Imposing $j_z = 0$

$$B_y = B_y(z) \quad H_z = \chi H_x \quad j_x = \sigma H_x$$

, try to find the one-dimensional problem (for variables parametrized in x):

$$1D \text{ Starting Model : } \begin{cases} \frac{d}{dz}(nu_{zi}) = n(\nu_{ion} - \nu_w) \\ nu_{zi} + n_n u_{zn} = const \\ m_i n u_{zi} \frac{du_{zi}}{dz} = -T_e \frac{dn}{dz} + j_x B_y - R_z - \nu_{ion} m_i n u_{zi} \\ \frac{dB_y}{dz} = -\mu_0 j_x \\ j_x = \sigma H_x \end{cases} \quad (5.38)$$

where

$$H_x(z) = -\frac{d\Phi}{dx} - u_{zi} B_y + \frac{T_e}{e} \frac{\partial \ln n}{\partial x} \quad \text{and} \quad \sigma = \frac{e^2 n}{m_e \nu_e}$$

A new quantity has been introduced, ν_w the wall plasma losses due to ionic flux at the sheaths of the anode and the cathode. Specifically,

$$n\nu_w(z) = |nu_{xi}|_{cathode} + |nu_{xi}|_{anode}$$

These values must be provided from studying the 'radial' profile. As this is not one of the scopes of this Bachelor's Thesis, the wall ionic losses are assumed to be constant and in the whole study performed equal to 0.

5.2.3 MPD: axial model with ionization

$$\frac{d}{dz}(nu_{zi}) = nn_n Q_{ion} - S_w \quad (5.39)$$

$$nu_{zi} + n_n u_{zn} = \frac{\dot{m}}{m_i A} = \Gamma_0 \quad (5.40)$$

$$m_i n u_{zi} \frac{du_{zi}}{dz} = -T_e \frac{dn}{dz} + j_x B_y - m_i n n_n (u_{zi} - u_{zn}) (Q_{cx} + Q_{ion}) \quad (5.41)$$

$$m_i n_n u_{zn} \frac{du_{zn}}{dz} = -m_i u_{zn} S_w - m_i n n_n (u_{zi} - u_{zn}) Q_{cx} \quad (5.42)$$

$$\frac{dB_y}{dz} = -\mu_0 j_x \quad (5.43)$$

$$H_z = -\frac{d\Phi}{dz} + \frac{T_e}{e} \frac{d \ln n}{dz} + u_x B_y \quad (5.44)$$

$$j_x = \sigma H_x \quad (5.45)$$

- $Q_{ion} = Q_{ion}(T_e)$ (ionization rate)
- S_w = sum of the local losses of ionic flux both at the anode and cathode. Similar to ν_w .
- $Q_{cx}(|u_i - u_n|)$ is the charge-exchange collision rate.
- $\nu_e = \nu_{en} + \nu_{ei} = n_n Q_{en} + n Q_{ei}$
- $\omega_{ce} = \frac{eB}{m_e}$

where $\sigma = \frac{e^2 n}{m_e \nu_e} = \frac{e^2}{m_e} \frac{n}{n Q_{ei} n_n Q_{en}}$

Equation 5.40 can be simplified to:

$$n_n = \frac{\Gamma - n u_{zi}}{u_{zn}} \quad (5.46)$$

Equation 5.44 is uncoupled from the system of equations and could be solved to determine the axial electric field $E_z = -\frac{\partial \Phi}{\partial z}$. In fact, this value has not been calculated since it implies that calculation of u_x from the 'radial profile'. Therefore, (5.44) will not be solved. However, it is known that $H_z = \frac{\omega_{ce}}{\nu_e} H_x$, so it is possible to calculate the "equivalent" axial electric field H_z .

Then, (5.39), (5.41), (5.42) and (5.43) constitute a system of ordinary differential equations in z for variables u_{zi} , u_{zn} , B and n .

(For the sake of simplicity, recall $u_{zi} \rightarrow u_i$ and $u_{zn} \rightarrow u_n$.)

$$\text{MODEL.v1} \quad \begin{cases} u_i \frac{dn}{dz} = nn_n Q_{ion} - S_w - n \frac{du_i}{dz} \\ nu_i \frac{du_i}{dz} = -c_s^2 \frac{dn}{dz} + \frac{\sigma}{m_i} (E_x - u_i B_y) B_y - nn_n (u_i - u_n) (Q_{cx} + Q_{ion}) \\ n_n u_n \frac{du_n}{dz} = -nn_n (u_i - u_n) Q_{cx} - u_n S_w \\ \frac{dB_y}{dz} = -\mu_0 \sigma (E_x - u_i B_y) \end{cases} \quad (5.47)$$

where $c_s^2 = \frac{T_e}{m_i}$: the sound speed.

Removing the singularity

The system has a removable singularity at $u_i = c_s$, or in other words, at $M = 1$ (being $M = \frac{u_i}{c_s}$, the Mach number). Substituting the 1st equation into the 2nd, the following equation is obtained after some algebra:

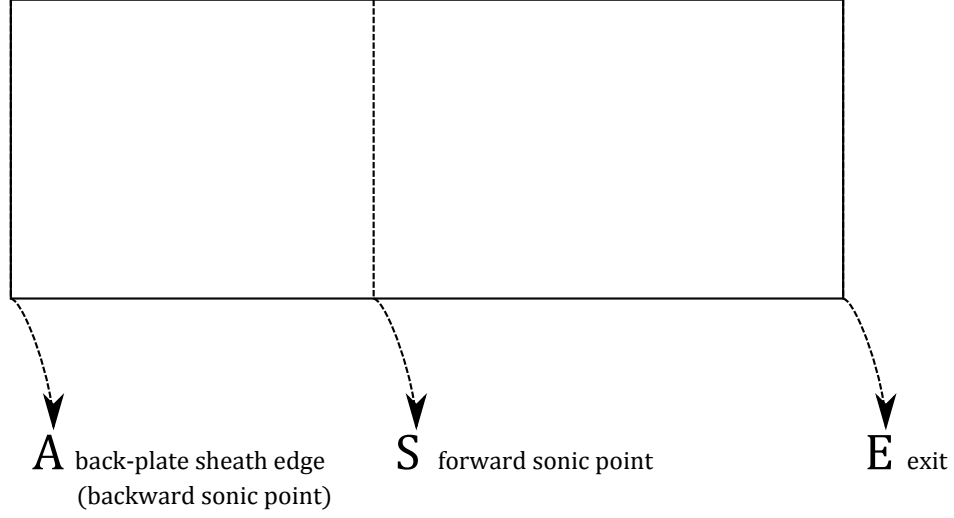
$$n(u_i^2 - c_s^2) \frac{du_i}{dz} = u_i \left[\frac{\sigma}{m_i} (E_x - u_i B_y) B_y - nn_n (u_i - u_n) (Q_{cx} + Q_{ion}) - \frac{c_s^2}{u_i} (nn_n Q_{ion} - S_w) \right] = G \quad (5.48)$$

The singularity can also be expressed in the differential equation for the plasma density by the procedure opposite to the one described before.

$$(u_i^2 - c_s^2) \frac{dn}{dz} = u_i (nn_n Q_{ion} - S_w) - \frac{\sigma}{m_i} (E_x - u_i B_y) B_y + nn_n (u_i - u_n) (Q_{cx} + Q_{ion}) \quad (5.49)$$

Let us assume that we know or we can compute from algebraic equations all parameters on the RHS of the four equations (including E').

Then, four **boundary conditions** are needed. These are located at A, S and E.



$$\begin{cases} \text{At } A, z = z_A : & u_n = u_{nA} \quad \text{and} \quad u_i = -c_s \\ \text{At } S, u_i = c_s : & G = 0 \\ \text{At } E, z = z_E : & B = 0. \end{cases}$$

This is a boundary-value problem with a regular sonic point at S. Runge-kutta methods can be used only if the integration is performed from point S to A and from S to E. In order to do so, the problem must be split in supersonic and subsonic regions. Also it is necessary to solve for the derivatives at point S.

Let us use an auxiliary variable ξ to normalize the problem stated in (5.47) :

$$\frac{d\xi}{dz} = \frac{1}{n(u_i^2 - c_s^2)}$$

We have now a set of 5 regular ordinary differential equations:

$$\text{MODEL.v2} \quad \left\{ \begin{aligned} \frac{dz}{d\xi} &= n(u_i^2 - c_s^2) \\ \frac{du_i}{d\xi} &= G \\ \frac{d\Gamma_i}{d\xi} &= \frac{dz}{d\xi} (nn_n Q_{ion} - S_w) \\ \frac{dB_y}{d\xi} &= -\frac{dz}{d\xi} \mu_0 \sigma (E_x - u_i B_y) \\ \frac{du_n}{d\xi} &= -\frac{dz}{d\xi} \frac{u_n S_w + nn_n (u_i - u_n) Q_{cx}}{n_n u_n} \end{aligned} \right. \quad (5.50)$$

This is a regular system of five ordinary differential equations which can be written as:

$$\frac{d\mathbf{y}}{d\xi} = \mathbf{f}(\mathbf{y}) \quad (5.51)$$

where $\mathbf{y}=\mathbf{y}(z, u_i, u_n, \Gamma_i, B_y)$.

However, $\mathbf{y}=\mathbf{y}_S$ is a stationary point where $\mathbf{f}(\mathbf{y}_S)=0$.

5.2.4 Determination of conditions close to S

$$\frac{d\mathbf{y}}{d\xi} = \mathbf{f}(\mathbf{y}) \quad \text{with} \quad \mathbf{f}(\mathbf{y}_S) = 0.$$

$$\mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{y}_S) + J\mathbf{f}(\mathbf{y}_S)\Delta\mathbf{y} + \dots \quad \Delta\mathbf{y} = \mathbf{y} - \mathbf{y}_S.$$

Therefore:

$$\frac{\Delta\mathbf{y}}{\Delta\xi} = J\mathbf{f}(\mathbf{y}_S)\Delta\mathbf{y}$$

Reaching to an eigenvalue problem

$$\left[J\mathbf{f}(\mathbf{y}_S) - (\Delta\xi)^{-1} \right] \Delta\mathbf{y} = \mathbf{0}$$

where

- $(\Delta\xi)^{-1}$: eigenvalues.
- $\Delta\mathbf{y}$: eigenvector subspace.

Then, the procedure can be summed up in:

1. Define / determine \mathbf{y}_S .
2. Compute $J\mathbf{f}(\mathbf{y}_S)$ numerically.
3. Compute the eigenvalues and eigenvectors. Determine quantitatively which is the physically valid one $\Delta\mathbf{y}_S$.
4. Choose an appropriate normalizing mechanism and apply the final $\Delta\mathbf{y}_S$ on initial sonic conditions at S.

This step defines which are the conditions from which the solver will start. Basically, the integration will be performed in both regions for $\xi \geq 0$, starting at two new points, S^+ and S^- .

Taking the set of equations 5.50, and changing the third equation by the equation for n , the following system is obtained.

$$\text{MODEL.v3} \quad \left\{ \begin{array}{l} \frac{dz}{d\xi} = n(u_i^2 - c_s^2) \\ \frac{du_i}{d\xi} = G = u_i \frac{\sigma}{m_i} B_y (E_x - u_i B_y) - \Gamma_i n_n (u_i - u_n) (Q_{cx} + Q_{ion}) - c_s^2 (nn_n Q_{ion} - S_w) \\ \frac{dn}{d\xi} = -n \frac{\sigma}{m_i} B_y (E_x - u_i B_y) + n^2 n_n (u_i - u_n) (Q_{cx} + Q_{ion}) + \Gamma_i (nn_n Q_{ion} - S_w) \\ \frac{dB_y}{d\xi} = -\frac{dz}{d\xi} \mu_0 \sigma (E_x - u_i B_y) \\ \frac{du_n}{d\xi} = -\frac{dz}{d\xi} \frac{u_n S_w - nn_n (u_i - u_n) Q_{cx}}{\Gamma - \Gamma_i} \end{array} \right. \quad (5.52)$$

$$\begin{aligned} \Gamma_i &= nu_i \\ n_n &= \frac{\Gamma - \Gamma_i}{u_n} \\ E_x &\approx \frac{\Delta V_{QN}}{L_x} \\ \Delta V_{QN} &= V_d - \Delta V|_{sheath} \\ c_s^2 &= \frac{T_e}{m_i} \\ \sigma &= \frac{e^2}{m_e} \frac{n}{nQ_{ei} + n_n Q_{en}} \end{aligned}$$

All collisional rates, $Q_{ion}, Q_{cx}, Q_{en}, Q_{ei}$, are functions of a single variable, T_e . They are also a function of the propellant properties but all computations made used Argon (Ar) as propellant. Therefore as the model is isothermal these variables are constant once T_e is selected. In fact Q_{cx} is a function of c_s . S_w is taken as a constant and also.

The new **boundary conditions** for the problem are:

At point S ($\xi = 0$): $z = 0$, $G = 0$ and $u_i = c_s$

At point E : $B = 0$ (this will be the main stopping condition for the Matlab code).

At point A : $u_i = -c_s$, $u_n = u_{nA}$ and $B = B_A = \frac{\mu_0 I_d}{L_y}$.

5.2.5 Nondimensionalization

1. Every dimensional analysis required firstly the definition reference parameters to define every quantity necessary to nondimensionalize the remaining variables.

In this problem, the reference mass flux $\Gamma_* = \Gamma$, the reference temperature is selected to be $T_* = T_e$ and the reference length the longitudinal scale of the engine $L_* = L_z$ are selected.

Therefore the remaining quantities can be properly calculated:

$$\bullet u_* = \sqrt{\frac{T_*}{m_i}}$$

$$\bullet n_* = \frac{\Gamma}{u_*}$$

$$\bullet \nu_* = \frac{u_*}{L_*}$$

$$\bullet Q_* = \frac{\nu_*}{n_*}$$

$$\bullet E_* = \frac{T_e}{eL_*}$$

$$\bullet B_* = \frac{E_*}{u_*}$$

$$\bullet j_* = \frac{B_*}{\mu_0 L_*}$$

$$\bullet \sigma_* = \frac{e^2}{m_i Q_*}$$

$$\bullet S_{w*} = n_* \nu_*$$

$$\bullet \xi_* = \frac{L_*}{n_* u_*^2}$$

2. The variables of the problem are:

$$\bullet \hat{z} = \frac{z}{L_*}$$

$$\bullet \hat{u}_j = \frac{u_j}{u_*}$$

$$\bullet \hat{n}_j = \frac{n_j}{n_*}$$

$$\bullet \hat{\Gamma}_j = \frac{\Gamma_j}{\Gamma_*}$$

- $\hat{b} = \frac{B_y}{B_*}$

(for all species j).

3. Non-dimensional parameters in the equations:

1. $\hat{e}_x = \frac{e\Delta V_{QN}}{T_e} \frac{L_z}{L_x}$.

2. $\hat{q}_{cx} = \frac{Q_{cx}}{Q_*}$

3. $\hat{q}_{ion} = \frac{Q_{cx}}{Q_*}$

4. $\hat{q}_{en} = \frac{m_e}{m_i} \frac{Q_{en}}{Q_*}$

5. $\hat{q}_{ei} = \frac{m_e}{m_i} \frac{Q_{ei}}{Q_*}$

6. $\hat{s}_w = \frac{S_w}{S_{w*}}$

The expressions to calculate the collisional rates are reasonable approximations for the purposes of this model. The rates of ionization, ion-electron collisions, electron-neutral collisions and ion-neutral collisions are:

$$Q_{ion}(T_e) = \sqrt{\frac{8T_e}{\pi m_e}} \sigma_{ion} \left(1 + \frac{T_e E_{ion}}{(T_e + E_{ion})^2} \right) \exp\left(\frac{-E_{ion}}{T_e}\right) \quad (5.53)$$

$$Q_{en}(T_e) = \sqrt{\frac{8T_e}{\pi m_e}} \sigma_{en} \quad (5.54)$$

$$Q_{ei}(T_e, n_e) = \left(\frac{T_e}{1eV} \right)^{-3/2} \ln \Lambda(T_e, n_e) 2.9 \times 10^{-12} \quad (5.55)$$

$$Q_{cx}(T_e) = \sqrt{\frac{T_e}{m_i}} \left(k_2 - k_1 \log_{10} \sqrt{\frac{T_e}{m_i}} \right)^2 \quad (5.56)$$

The value of $\ln \Lambda$ is assumed to be 9 as the temperatures are in the order of 2-5 eV. This quantity has been defined in previous chapters, specifically after (4.5). The other constants involved in the previous definitions are propellant dependent. For the case of Argon, they are $E_{ion} = 15.76eV$, $\sigma_{ion} = 2.8 \times 10^{-20}m^2$, $\sigma_{en} = 1.5 \times 10^{-20}m^2$, $k_1 = 10.5 \times 10^{-10}m$, $k_2 = 1.67 \times 10^{-10}m$.

5.3 Complete Axial Model: final version

All variables and boundary conditions expressed in this model are nondimensional. Let us drop the hats from the variables.

$$\text{MODEL} \left\{ \begin{array}{l} \frac{dz}{d\xi} = n(u_i^2 - 1) \\ \frac{du_i}{d\xi} = G = u_i \sigma b(e_x - u_i b) - \Gamma_i n_n (u_i - u_n)(q_{cx} + q_{ion}) - (n n_n q_{ion} - s_w) \\ \frac{dn}{d\xi} = -n \sigma b(e_x - u_i b) + n^2 n_n (u_i - u_n)(q_{cx} + q_{ion}) + \Gamma_i (n n_n q_{ion} - s_w) \\ \frac{db}{d\xi} = -\frac{dz}{d\xi} \mu_0 \sigma_* L_* u_* \sigma(e_x - u_i b) = -\frac{dz}{d\xi} Rm^* \sigma(e_x - u_i b) \\ \frac{du_n}{d\xi} = -\frac{dz}{d\xi} \frac{u_n S_w - n n_n (u_i - u_n) q_{cx}}{\Gamma - \Gamma_i} \end{array} \right. \quad (5.57)$$

$$\text{where} \left\{ \begin{array}{l} \Gamma_i = n u_i \\ n_n = \frac{1 - \Gamma_i}{u_n} \\ \sigma = \frac{n}{n q_{ei} + n_n q_{en}} \\ Rm^* = \mu_0 \sigma_* u_* L_* \end{array} \right. \quad (5.58)$$

Rm^* is again the magnetic reynolds number computed from the parameters of reference T_e , Γ and L_z .

5.4 The Matlab Code

Overview

The script is a modeler of the dynamics and performance of an cylindrical shaped MPDT. Given as inputs the geometry of the engine, the electron temperature, and the gas used as propellant, it is able to estimate the global performance parameters of an engine given the initial conditions estimated at the sonic point. This model is a first step in the development of a more realistic procedure as passing parameters to the thruster. The code returns the axial evolution of the variables of the problem and some figures as the efficiency, thrust, power consumed, exhaust speed, voltage and current used.

Structure

The matlab code for this model has several scripts:

- COMPLETE_AXIAL_MODEL: the main script
- com_model_ode: the function containing the system of ordinary differential equations
- The event functions for both subsonic and supersonic regions (event_function_subsonic and event_function_supersonic).
- f_model: the function to evaluate the derivatives in order to calculate the jacobian matrix.

Description

The constants of the problem are defined. The next step is to define the geometry of the engine. The axial length L_z and the radius of anode r_a and cathode r_c . Then the geometry is transformed in a similar problem with squared channel, so L_x and L_y are now introduced. Afterwards the code requires to define the gas to be used as propellant and to include its atomic mass in m_{prop} , and also the electron temperature. Furthermore, the code requires to specify the collisional rate constants for that gas species, using (5.53),(5.54),(5.55) and (5.56).

The main quantities of the problem are calculated to later redimensionalize the problem. This is also necessary to provide the required vector of parameters to ode45. The parameters provided are given in a vector called $P_{opt} = [e_{xS} q_{ion} q_{en} q_{cx} q_{ei} s_w Rm^*]$. These are all the nondimensional parameters of the problem. Notice that e_{xS} is the radial electric field at the whole thruster as it is considered to be constant. It is calculated from the condition $G = 0$ at the sonic point.

Afterwards it is necessary to define the conditions at the sonic point. Sonic conditions are defined in a 1×5 vector : $S_0 = [z_S u_{iS} n_{iS} b_S u_{nS}]$. In fact, only three conditions vary from one case to another, as $z_S = 0$ and $u_{iS} = 1$ by definition. These values are:

1. $\Gamma_{iS} = n_{iS}$: the ionic flux at the sonic point.
2. u_{nS} : the neutrals speed at the sonic point .
3. b_S : the magnetic field at the sonic point .

Using S_0 it is possible to define the vectors S_0^+ and S_0^- by solving the mathematical problem stated in subsection 5.2.4. Using function `f_model`, the jacobian matrix can be numerically calculated. A first order interpolation of the slopes is used to define the derivatives. A 5×5 matrix is defined with increments of +0.1% and -0.1% of each variable. Then the derivatives in all directions can be calculated (the Jacobian Matrix), and finally the code solves for the increments limiting the maximum increment to be a 5% of the value of that condition at the sonic point. This is the normalizing factor applied on the vector of initial conditions. This normalizing factor also takes into account the sign of the first derivative of the system, which is $\frac{dz}{d\xi}$. As the starting point for both iterations is the sonic point, the value of this derivative must be:

$$\begin{cases} \text{Supersonic region : } \frac{dz}{d\xi} > 0 \\ \text{Subsonic region : } \frac{dz}{d\xi} < 0 \end{cases}$$

The scheme of the Jacobian Matrix calculation is shown hereinafter:

$$Y_{S+} = \begin{pmatrix} z_s + \Delta z & u_{iS} & n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} + \Delta u_{iS} & n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} + \Delta n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} & b_S + \Delta b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} & b_S & u_{nS} + \Delta u_{nS} \end{pmatrix}$$

$$Y_{S-} = \begin{pmatrix} z_s - \Delta z & u_{iS} & n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} - \Delta u_{iS} & n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} - \Delta n_{iS} & b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} & b_S - \Delta b_S & u_{nS} \\ z_s & u_{iS} & n_{iS} & b_S & u_{nS} - \Delta u_{nS} \end{pmatrix}$$

Then, using these to matrices and combining them with use of `f_model` function, it is possible to calculate all the derivatives and so the Jacobian Matrix $\mathbf{J}\bar{\mathbf{f}}$ by:

$$\mathbf{J}\bar{\mathbf{f}} = \begin{bmatrix} \frac{\partial \bar{\mathbf{f}}}{\partial y_1} & \frac{\partial \bar{\mathbf{f}}}{\partial y_2} & \frac{\partial \bar{\mathbf{f}}}{\partial y_3} & \frac{\partial \bar{\mathbf{f}}}{\partial y_4} & \frac{\partial \bar{\mathbf{f}}}{\partial y_5} \end{bmatrix}$$

where,

$$J(f)_{ij} = \frac{\partial f_i}{\partial y_j} = \frac{f_i(y_{jS} + \Delta y_j) - f_i(y_{jS} - \Delta y_j)}{2\Delta y_j}$$

Note that i stands for row and j stands for column.

Then both integrations are performed. Two stopping functions are used, one for each region. The **stopping conditions** for both regions are:

SUBSONIC REGION

1. Stop when $z=10$. This value is due to the fact that the iteration could not converge. If z goes to too high values the code stops the iteration.
2. Stop when $u_i = -1$. This is due to the fact that at the backplate there is a negative sonic point or ionic flow going in negative direction. That is one of the boundary conditions assumed for the subsonic region.
3. Stop when $u_n = 0.000001$. To avoid errors when $u_n = 0$ the runner stops when $u_n \approx 0$.

SUPERSONIC REGION

1. Stop when $b=0$. This is the condition of the magnetic field at the chamber exit is null. This condition has a high sensitivity to changes in the initial conditions. Sometimes the code does not converge since this condition is not fulfilled and the magnetic field behaves as if $Rm \rightarrow \infty$.
2. Stop when $z=100$. This value is due to the fact that the iteration could not converge. If z goes to too high values the code stops the iteration.

Program performance

The main limitation of the code is the high sensitivity of both subsonic and supersonic solutions to the initial conditions imposed. An exhaustive procedure has been performed to understand the difficulties that arised in the iteration.

Basically, when the three conditions are increased up to some combination of them, the solution is for $Rm \approx \infty$ or flat magnetic field b . In other words b does not decay in the whole acceleration process and sometimes even increased, due to a possible numerical error.

As a consequence, the procedure to obtain desired cases is iterative and based on balancing the conditions in order to adjust the desired L_z , and an appropriate boundary condition at subsonic region (at point A). Also high values of ionic flux at the exit are desired. All these conditions change with T_e and the other problem parameters. Moreover these conditions seem to be difficult to fulfil at low temperatures ($T_e \leq 2$). In next section, the main interesting cases found are detailed.

The program does not take long computational times when the solution is obtained for both regions. In case no solution is achieved the code is solution is interrupted by the maximum z stopping conditions any of the two regions.

5.5 Results

The code is very sensitive to changes on the sonic conditions imposed based on other problem parameters. Therefore, in order to find the operation of an engine with the required longitudinal dimension (that matches the one studied), an iteration process must be made adjusting these three conditions at S. As has been explained in previous chapters, after a deep bibliography review concerning MPD thrusters operation, the normal range of values for the electron temperature inside the plasma produced in these thrusters is 2-5 eV. Therefore 4 cases to satisfy the geometry of the engine for 2, 3 4 and 5 eV have been studied. T_e affects directly the collisional rates and therefore the behaviour of the system of ordinary differential equations. That dependance has been analyzed and the results are shown hereafter.

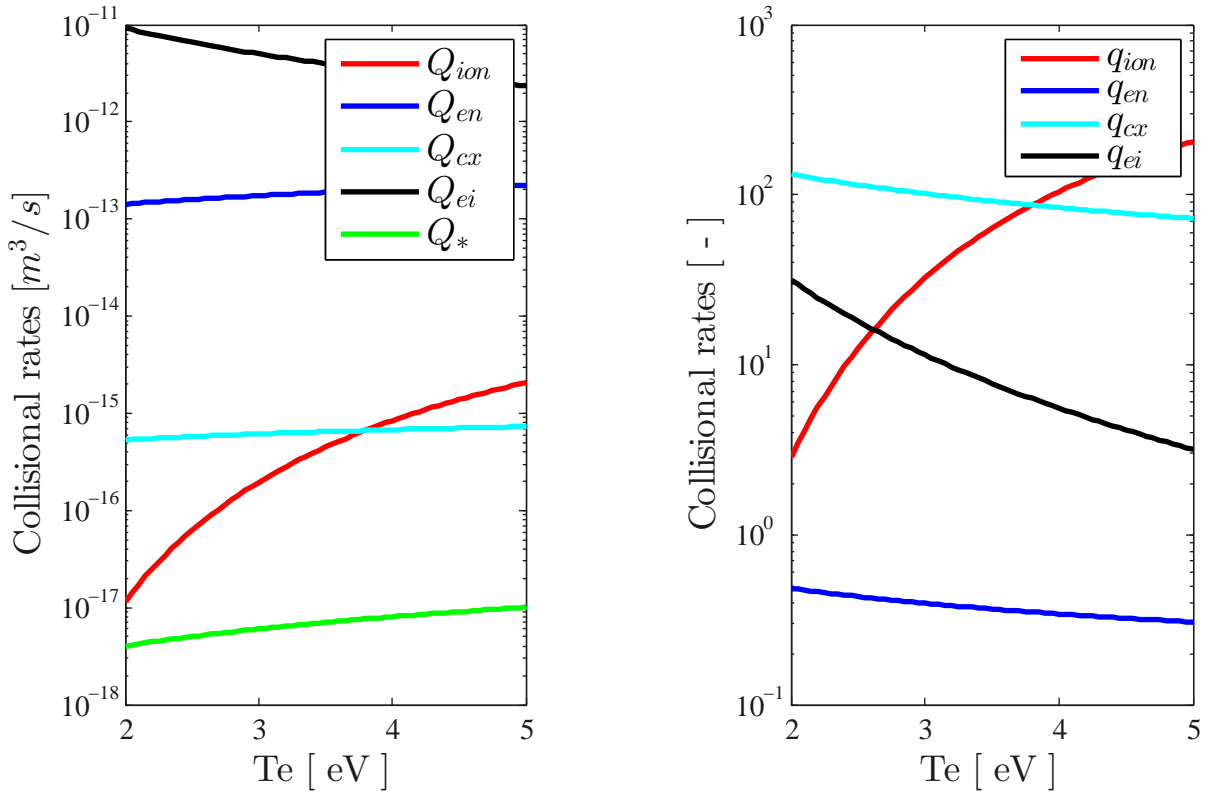


Figure 5.1: Collisional rates dependances as a function of T_e for Ar species.

As can be seen all collisions are nondimensionalized with respect to the reference collisional rate Q_* in Fig.5.1. Looking at this figure, it can be seen that the collisional process of charge exchange between neutrals and ions is high. Also, it can be noticed that the ionization rate increases much with T_e and opposite to electron-ion collisions, electron-neutral collisions and collisional exchange. In fact only Q_{ei} decreases with T_e but in the model as Q_* changes also, the net change of q_{en} and q_{cx} is negative with decreasing the temperature. This fact suggests that the higher the temperature, the faster the ionization process occurs in the channel.

5.5.1 Case analysis

Notice that all the cases studied are for the following operational parameters:

L_Z [m]	r_a [m]	r_c [m]	Propellant	Γ [$kg/m^{-2}s$]
0.1	0.05	0.01	Argon	0.7958

Table 5.1: Engine operational parameters for all cases studied

Also, the cases enhanced in this document present approximately the same induced magnetic field, in the order of 1500G. The purpose is to be able to discuss the results and compare them to extract conclusions, so the higher the amount of similar operational parameters between cases, the more interesting the analysis.

The obtention of these cases is based on the seek of several boundary conditions and behavior with the manipulation of the sonic conditions. The intention is to fulfil every boundary condition present in both regions. It is sought for backward ionic flux in the backplate (sonic if possible). Also, the ionic flux at the exit should be as high as feasible, in order to assure that the plasma accelerated is almost completely ionized. This fact will increase the performance of the thruster. Last but not least, the magnetic field should vanish at the exit of the chamber.

An interesting stiffening of the ordinary differential system of equations was found as soon as the T_e of the case was decreased. In fact, the model had many difficulties to describe the subsonic region in those conditions.

Not only the subsonic region solution was highly sensitive to the changes in the sonic conditions. In fact the supersonic solution had a similar issue when a maximum of the conditions was achieved. The determination of that maximum relationship between the conditions. However the cases showed a behaviour similar to the acceleration model when magnetic Reynolds were to high. The channel length became wider and the magnetic field equilibrated the electrostatic field in most of the channel. This behavior was obtained for the acceleration model when Rm was increased until too high values, and the solution of \bar{E} became almost impossible to obtain.

Starting from the case of $T_e = 5$, it is important to notice that in the thruster behaves as expected. The magnetic field and the ions velocity increases similarly to the acceleration model prediction. It can be noticed that almost the whole flow is ionized. The magnetic field vanishes at the exit and what is very interesting is that both ions and neutrals obtain nearly the same exhaust speed. As soon as collisions are increased, or T_e is lowered, this phenomena becomes clearer.

The acceleration process shows a dependance on the Rm . As for the acceleration model, the highest Rm showed the fastest acceleration and two localized regions where the current is concentrated, one at the chamber entrance and one at the exit. This change in the shape of the acceleration process is dependent on the T_e as was the value changed. Results show that the highest T_e is, the better the acceleration process. The different cases studied are shown hereinafter.

Case 5 eV

Nondimensional neutrals speed u_{nS} [-]	0.613
Nondimensional ionic flux g_i [-]	0.84
Nondimensional magnetic field bS [-]	11.9

Table 5.2: Sonic conditions for 5 eV case.

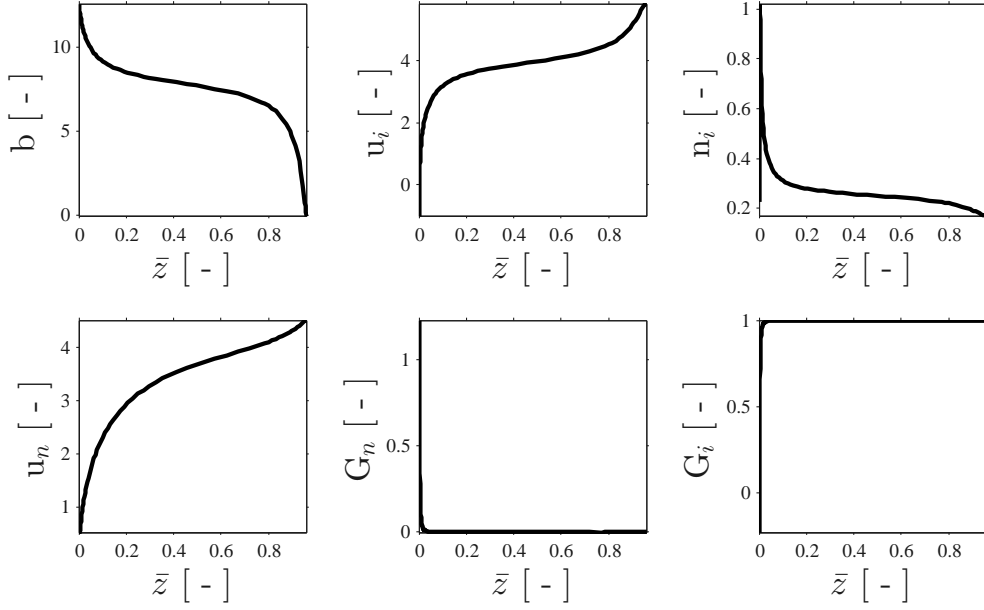


Figure 5.2: Complete solution for 5 eV case.

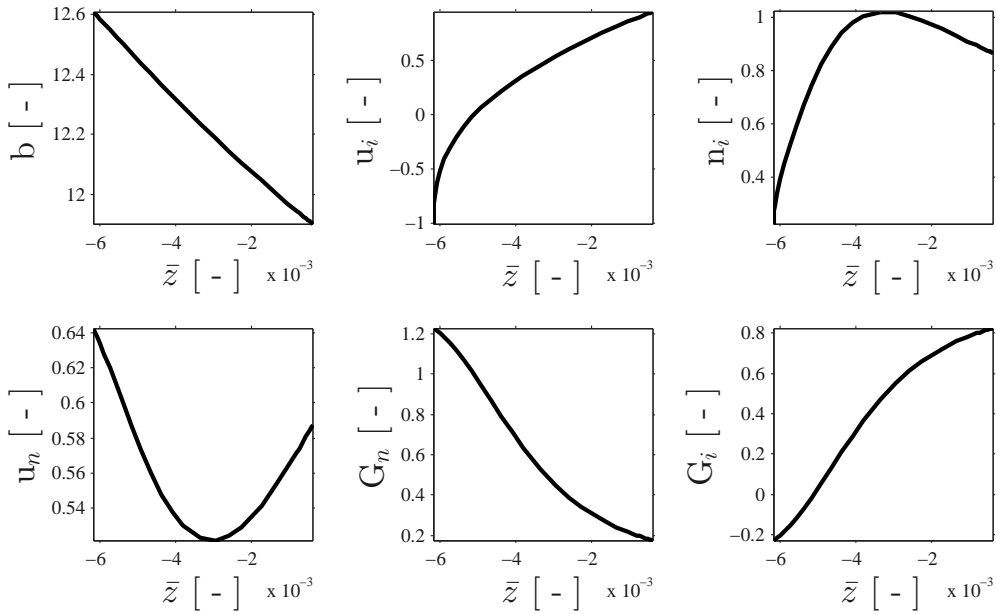


Figure 5.3: Subsonic region for 5 eV case.

Case 4 eV

Nondimensional neutrals speed u_{nS} [-]	0.56245
Nondimensional ionic flux g_i [-]	0.87
Nondimensional magnetic field bS [-]	11.9

Table 5.3: Sonic conditions for 4 eV case.

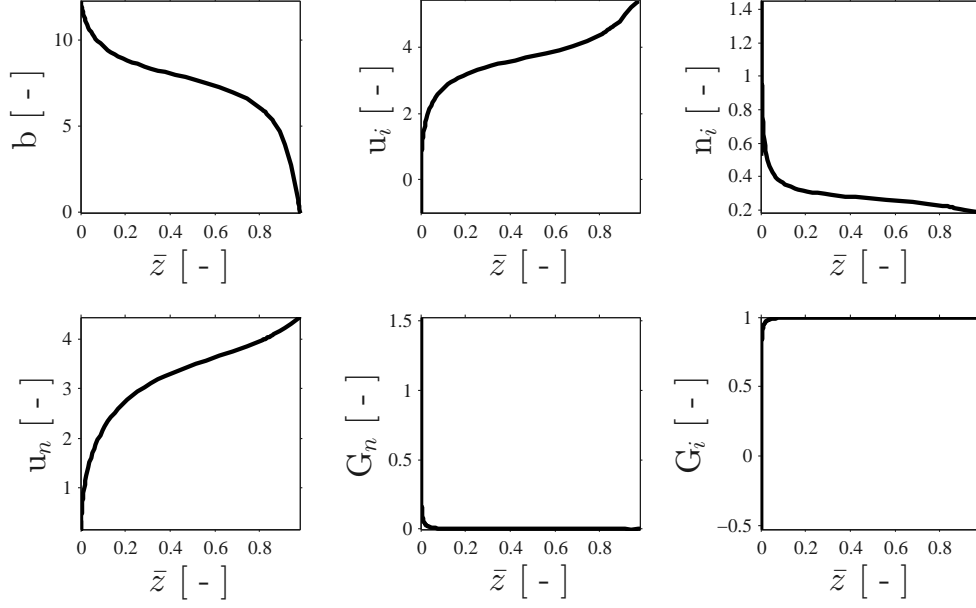


Figure 5.4: Complete solution for 4 eV case.

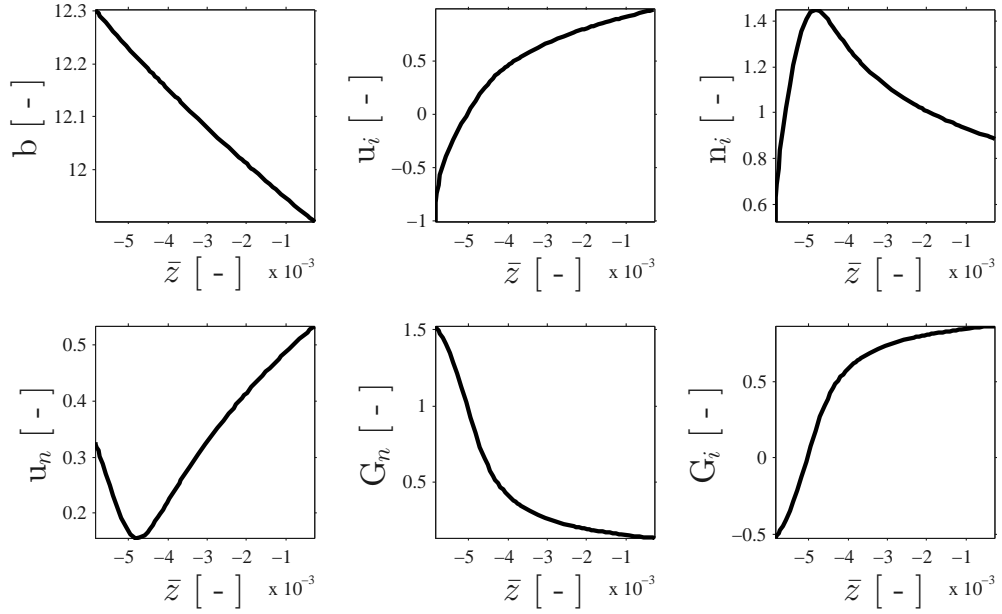


Figure 5.5: Subsonic region for 4 eV case.

Case 3 eV

Nondimensional neutrals speed u_{nS} [-]	0.66
Nondimensional ionic flux g_i [-]	0.625
Nondimensional magnetic field bS [-]	13.8

Table 5.4: Sonic conditions for 3 eV case..

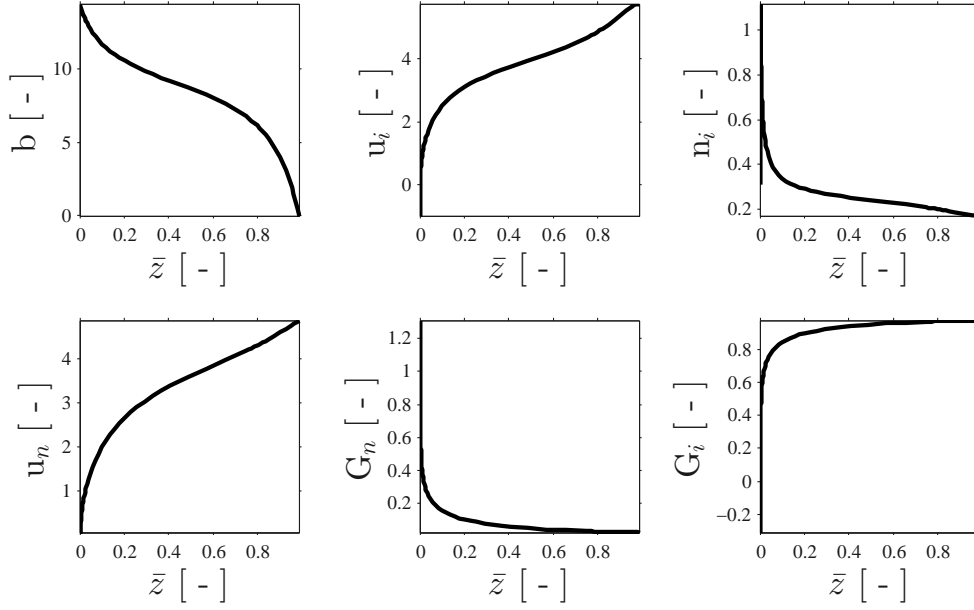


Figure 5.6: Complete solution for 3 eV case.

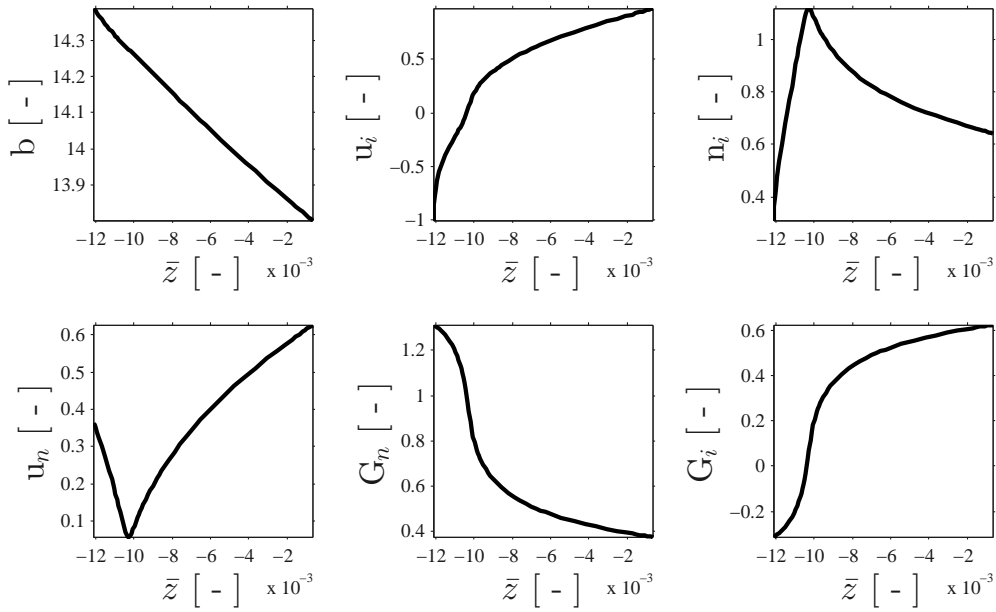


Figure 5.7: Subsonic region for 3 eV case.

Case 2 eV

Nondimensional neutrals speed u_{nS} [-]	0.75
Nondimensional ionic flux g_i [-]	0.7
Nondimensional magnetic field bS [-]	13

Table 5.5: Sonic conditions for 2 eV case.

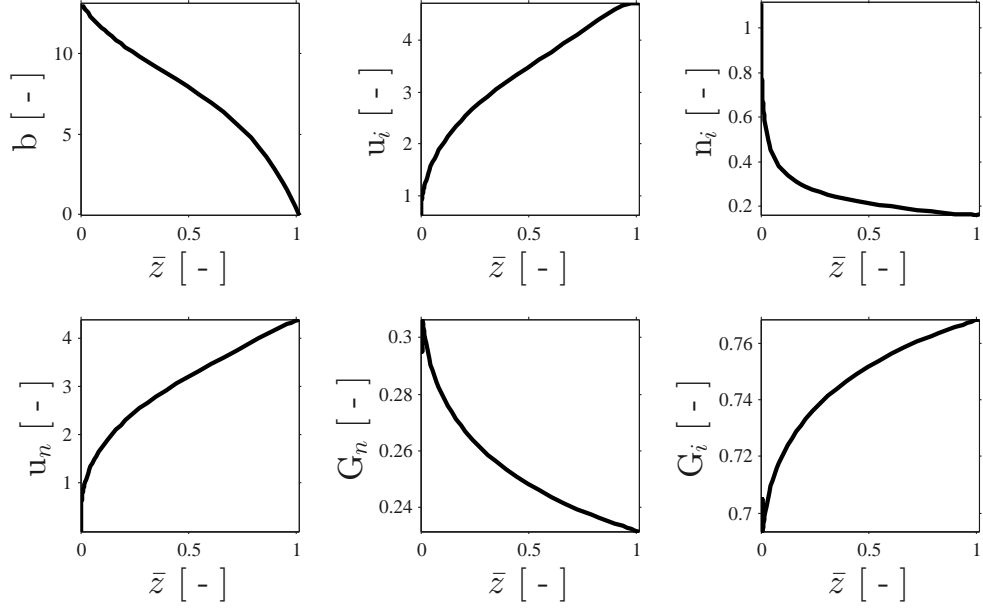


Figure 5.8: Complete solution for 2 eV case.

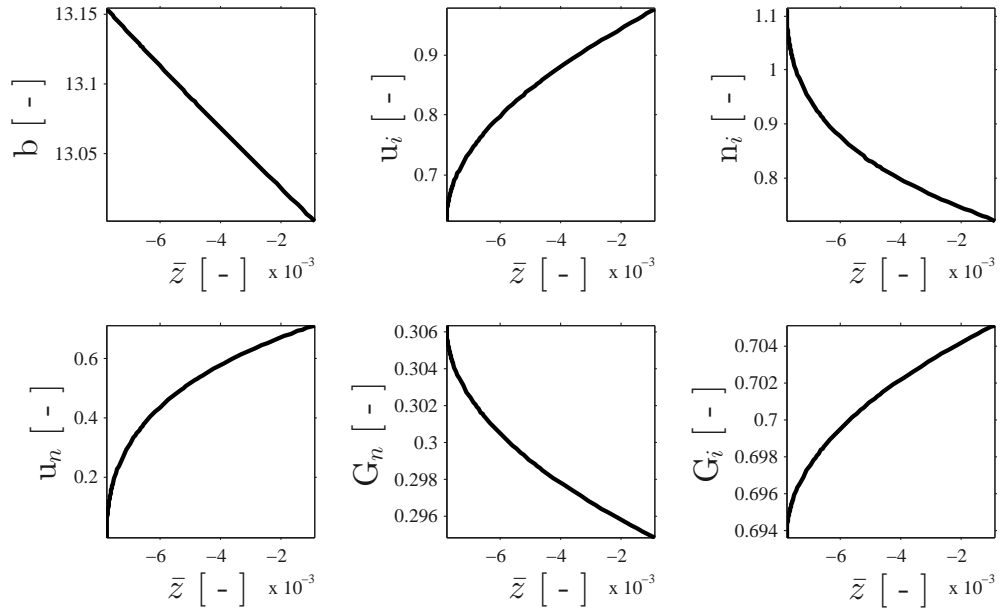


Figure 5.9: Subsonic region for 2 eV case.

5.5.2 Operational performance and subsonic region

Case (T_e)	2 eV	3 eV	4 eV	5 eV
F [N]	57.4172	92.6	100.94	122.01
η_t [%]	62.71	66.41	74.74	72.87
P_{req} [W]	4.38×10^5	1.08×10^6	1.14×10^6	1.7×10^6
P_{jet} [W]	2.74×10^5	7.14×10^5	8.49×10^5	1.24×10^6
E_x [V/m]	673	1118	1196	1564
V_d [V]	26.91	44.7	47.83	62.57
I_d [A]	1.63×10^4	2.41×10^4	2.38×10^4	2.72×10^4
u_E [km/s]	9.57	15.43	16.82	20.33
I_{sp} [s]	976	1573	1715	2073
B_o [T]	0.1085	0.1603	0.1583	0.1814
l_z [m]	0.101	0.0988	0.098	0.0958
U_{nA} [m/s]	0.002	983	1014	2233
U_{iA} [m/s]	1892	-2692	-3108	-3475
u_{iA} [-]	0.86	-1	-1	-1
g_{iA} [-]	0.9342	-0.31164	-0.5239	-0.2262
g_{iE} [-]	0.95574	0.9796	0.9998	≈ 1

Table 5.6: Performances and obtained parameters of the cases studied

Table 5.6 expresses the whole numerical information obtained from the most relevant cases obtained, in a compact form. Notice that the cases for 3 4 and 5 eV of T_e , obtain desired values for the boundary conditions both at the exit and at the entrance. Negative ionic flux is obtained at the backplate and positive and almost equal to 1 at the exit of the thruster chamber.

As can be seen in table 5.6 the complete axial model proves that MPD thrusters operation is much better in the electromagnetic regime, where the power consumed is high. This fact is proven by the increase on the efficiency, the specific impulse, the thrusts and all figures of merit of the thruster.

The ions have been proved to be able to reach negative sonic speed at the backplate. This condition, can be clarified by the following explanation. When a neutral enters the chamber at point A is ionized quickly. The electron and the ion velocities can be estimated by the kinetic energy the have. Therefore, the electrons speed is of the order of $\sqrt{\frac{T_e}{m_e}}$. The ions speed on its side, is of the order of $\sqrt{\frac{T_e}{m_i}}$. Both motions produce a corresponding current, an electron current $j_e = en_e u_e$ and an ionic current $j_i = en_i u_i$. Assuming that the plasma quasineutral ($n_e \approx n_i$) then:

$$\frac{j_i}{j_e} \sim \frac{en_i u_i}{en_e u_e} \sim \sqrt{\frac{m_e}{m_i}} \sim 10^{-4}$$

Therefore the current of the electrons is much higher than the ions current. This

explains the unsteady process which is based on the fact that the backplate is a dielectric and there, the dielectric wall tends to attract quicker the electrons than the ions. Therefore a strong negative potential is created close to the wall, inducing a backwards motion of the ions.

After a transient, the steady solution to this process is the one shown as results in cases for T_e 3 eV, 4 eV and 5 eV. The electrons are no longer attracted to that region, and what happens is that neutrals that enter the steady operation plasma, see an ionic flux opposing their motion. These neutrals are gradually ionized and decelerated until specific point. This point can be defined as a 'source' of ionic species, as can be noticed from the ions density and ionic flux figures. Another interpretation is that this point is the transition point from the ionization dominated region to the acceleration dominated region. As far as this critical point is overcome by the neutrals, they start to accelerate with the ions and continue ionizing throughout the whole thruster chamber.

This mechanism may be difficult to model at high densities, or at highly collisional plasmas. The thickening of this interesting region as long as the temperature is increased suggest that low temperature plasmas do not follow this behavior and possibly other theoretical model for this region should be applied.

Finally, let us calculate the Debye length of the backplate sheath for all cases discussed that have a negative ionic flux at point A. The debye length λ_D can be calculated as has been explained in previous chapters just after the quasineutrality definition. Equating the potential energy to the thermal energy of the electrons the definition of the debye length arises.

Knowing that $\nabla \cdot \mathbf{E} = \frac{e(n_i - n_e)}{\varepsilon_0}$ and recalling the electrostatic potential definition $\mathbf{E} = -\nabla\Phi$, then

$$-\nabla^2\Phi = \frac{e(n_i - n_e)}{\varepsilon_0}$$

If we take the order of magnitude of the quantities involved in the latter equation,

$$\frac{\Delta\Phi}{L^2} \approx \frac{en_o}{\varepsilon_0}$$

.

Assuming that both the potential energy and the thermal energy of electrons are comparable

$$e\Delta\Phi \sim T_e k_B$$

the Debye characteristic length appears as the scale at which the former condition is satisfied.

$$\lambda_D = \sqrt{\frac{\varepsilon_0 T_e k_B}{e^2 n_o}} \quad (5.59)$$

Typically $\lambda_D \ll L$, and this voltage structures appear at the walls of the thruster. These voltage sheaths are usually treated as boundary layers as mentioned before. When the plasmas present lower densities these sheaths can extend as does the ionization dominated region.

Then, using the values for T_e and n_o which is assumed to be of the order of the ions density at the entrance n_{iA} , the different values of the debye length λ_D are found to be:

Case (T_e)	3 eV	4 eV	5 eV
$n_o [m^{-3}]$	1.3889×10^{21}	2.022×10^{21}	7.81×10^{20}
$\lambda_D [m]$	3.45×10^{-7}	3.31×10^{-7}	5.95×10^{-7}

Table 5.7: Debye lengths for the practical cases

5.6 Comparison with the Acceleration Model

In this section, the complete axia model and the acceleration model are compared to each other.

Let us start with the importance of the electrothermal acceleration or the pressure gradient in generalized plasma momentum equation. Recall (3.9). Assume that the pressure gradient produces considerable momentum gain on the plasma, then both terms are comparable.

$$\frac{\Delta p}{\rho \mathbf{u} \cdot \nabla \mathbf{u}} = \frac{\nabla n T}{m_i n \mathbf{u} \cdot \nabla \mathbf{u}} \sim \frac{T}{m u^2} \sim \frac{T_e}{m_i u^2} \sim \frac{c_s^2}{u^2} = \frac{1}{M^2}$$

As the model is isothermal, the pressure gradient is equal to the product of the temperature and the particle density gradient. Then assuming that the gradients of velocity and particle density are in the same scale, the Mach number appears in this comparison. This implies that as long as the mach is of order unity the pressure gradient contributes to the acceleration process. When $M \gg 1$ the pressure terms can be neglected from the momentum equation. This analysis suggests that the acceleration process of the plasma is dominated by the Lorentz force as long as the flow accelerates beyond the sonic point. In that region, the flow resulting from the complete axial model should behave roughly as the acceleration model predicts. Notice that the ionization process has mostly happened in the subsonic region in all of the practical cases shown in the results, so it is another argument to support the former reasoning.

In order to further analyze this, let us obtain that information in the theoretical behavior of the velocity looking at the onedimensional momentum equation for ions in a fully ionized plasma.

$$m_i n u \frac{du}{dz} = j B - T_e \frac{dn}{dz}$$

Using Ampere's law for onedimensional flow

$$\frac{dB}{dz} = -\mu_0 j$$

the momentum equation can be expressed as

$$m_i n u \left(1 - \frac{c_s^2}{u^2} \right) \frac{du}{dz} = -\frac{B}{\mu_0} \frac{dB}{dz}$$

The latter equation can be expressed in terms of the Mach number M and the mass flux Γ_0

$$\Gamma_0 \left(1 - \frac{1}{M^2} \right) \frac{du}{dz} = -\frac{B}{\mu_0} \frac{dB}{dz}$$

Integrating with respect to z , the following relationship is obtained,

$$\Gamma_0 u \left(1 - \frac{1}{M^2} \right) = -\frac{B^2}{2\mu_0} + \text{constant}$$

For $M \gg 1$, the relationship between the magnetic field and the velocity is

$$u^2 \Gamma_0 + \frac{B^2}{2\mu_0} = \text{constant}$$

Taking into account that not only most of the channel is supersonic ($M < 1$) but also that the subsonic region is close to 1% of the total channel length in all cases, then the ideal model seems to be an appropriate estimate for the acceleration process. Furthermore, this argument is supported also by the fact that collisions dominate in the subsonic region as well, where nearly all the ionization process takes place.

Chapter 6

Conclusions

The parametric study performed in the Acceleration Model revealed that MPD thrusters operate more efficiently at high T_e . Also, it outlined that the mass flow rate has a direct effect on the efficiency and is nearly proportional to the power required. Furthermore this study showed that, as experimental data proves, self-field MPD thrusters start to have efficient operation in the tenths of Megawatt range.

In fact, both models have shown that MPD thrusters operate more efficiently in the electromagnetic regime, or in other words, at high powers. However an important contribution to the acceleration process appears from the pressure gradient as long as the mach number is not high.

Moreover, the models explained in this Bachelor Thesis have outlined that at higher currents, other thruster performance parameters as the propulsive efficiency, the thrust delivered, and the specific impulse increase.

It has been concluded that the ionization process is more complicated to study the lower the T_e . Decreasing T_e means that the plasma studied is colder and more collisional, which difficults the capability of the Complete Axial Model to obtain results with $u_{iA} < 0$ and G_{nA} in the region $T_e \leq 2\text{ eV}$.

The use of the Complete Axial Model has proved that the subsonic region in MPD thrusters is typically much smaller than the supersonic region. Subsonic regime is mostly dominated by ionization processes as can be seen at the different cases shown for $T_e = 3, 4$ and 5 eV . Therefore nearly the whole ionization process occurs in a very thin layer at the chamber entrance of length of the order of 1% of the full axial scale of the thruster chamber.

Concerning the comparison between models, not only most of the channel is supersonic ($M < 1$) in all cases but also the subsonic region is close to 1% of the total channel length. Taking into account that both the ionization process and the electrothermal acceleration play only a significant role in the subsonic regime, it can be concluded that the ideal model or acceleration model seems to be an appropriate estimate for the acceleration process and the operational performance of self-field magnetoplasmady-

namic thrusters.

Also, it is important to remark that the magnetic Reynolds number Rm has been seen as an important parameter in the performance of axial engines, not only in two dimensional models. This fact has been proved by both models developed, in which this parameter appeared.

Finally it must be enhanced the fact that the acceleration model with ionization overcomes some difficulties of the equations studied, like the sonic singularity by further complicated the procedure. Although the procedure is intricate, accurate and realistic solutions can be obtained by this rigorous manipulation of plasma acceleration equations.

Future Prospects

Future work will focus on the complete development of a simulation 2D code modelling the plasma discharge in a SF-MPDT. The code should be based on an axisimetric steady plasma model. Either the development of a new 2D model or the implementation of attachment functions that introduce the radial problem, are equivalent options.

Variable cross-sectional area could be introduced in the problem. Different nozzle geometries could be tested as convergent-divergent nozzle.

All plasma production, plasma acceleration and also the electrostatic sheaths at the walls should be modelled. Furthermore hall effect is not negligible and makes the current non-parallel to the electric field. Also, total energy equation should be formulated.

Bibliography

- [1] <http://www.jpl.nasa.gov/>.
- [2] <http://www.esa.int>.
- [3] <http://www.nasa.gov/>.
- [4] Eduardo Ahedo. Plasmas for space propulsion. *Plasma Physics and Controlled Fusion*, 53(12):124037, 2011.
- [5] Mariano Andrenucci. Magnetoplasmdynamic thrusters. *Encyclopedia of Aerospace Engineering*.
- [6] M Auweter-Kurtz and H Kurtz. High thrust density electric propulsion for heavy payload in-space transportation. In *4th International Spacecraft Propulsion Conference*, volume 555, page 126, 2004.
- [7] Ministerio de Empleo y Seguridad Social BOE. Xvii convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos. <https://www.boe.es/boe/dias/2013/10/25/pdfs/BOE-A-2013-11199.pdf>. Published the October 25th, 2013.
- [8] Edgar Y Choueiri. New dawn for electric rockets. *Scientific American*, 300(2):58–65, 2009.
- [9] Edgar Y Choueiri and Hideo Okuda. Anomalous ionization in the mpd thruster. In *23rd International Electric Propulsion Conference, Seattle, WA, USA*, 1993.
- [10] EY Choueiri and JK Ziemer. Quasi-steady magnetoplasmdynamic thruster performance database. *Journal of Propulsion and Power*, 17(5):967–976, 2001.
- [11] Robert G. Jahn Edgar Y. Choueiri. Electric propulsion. *Encyclopedia of Physical Science and Technology, Third Edition*, 2002.
- [12] Dan M Goebel, Ira Katz, John Ziemer, JR Brophy, JE Polk, and Lee Johnson. *Electric propulsion research and development at JPL*. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2005.
- [13] OA Gorshkov, VN Shutov, KN Kozubsky, VG Ostrovsky, and VA Obukhov. Development of high power magnetoplasmdynamic thrusters in the ussr. In *30th International Electric Propulsion Conference, Florence, Italy. IEPC-2007-136*, 2007.

- [14] Robert G Jahn. *Physics of electric propulsion*. Courier Dover Publications, 2012.
- [15] AD Kodys and EY Choueiri. A critical review of the state-of-the-art in the performance of applied-field magnetoplasmadynamic thrusters. In *The 41st Joint Propulsion Conference (JPC)*, 2005.
- [16] H Maecker. Plasma jets in arcs in a process of self-induced magnetic compression. *Zeitschrift fur Physik*, 141(1):198–216, 1955.
- [17] M Martinez-Sanchez and James E Pollard. Spacecraft electric propulsion-an overview. *Journal of Propulsion and Power*, 14(5):688–699, 1998.
- [18] EJ Sheppard. Ionization rate models and inlet ignition in self-field mpd thrusters. *Sea*, 10000(20000):40000, 1993.
- [19] Prof. Manuel Martinez Sánchez. Lecture 21: Electrostatic versus electromagnetic thrusters. <http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-522-space-propulsion-spring-2004/lecture-notes/lecture21.pdf>. 16.522, Space Propulsion.
- [20] Prof. Manuel Martinez Sánchez. Lecture 22: A simple model for mpd performance-onset. <http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-522-space-propulsion-spring-2004/lecture-notes/lecture22.pdf>. 16.522, Space Propulsion.
- [21] George P Sutton and Oscar Biblarz. *Rocket propulsion elements*. John Wiley & Sons, 2010.
- [22] K Toki, Y Shimuzu, and K Kuriki. Electric propulsion experiment (epex) of a repetitively pulsed mpd thruster system onboard space flyer unit (sfu). In *International Electric Propulsion Conference, IEPC*, pages 97–120, 1997.

Appendix A

Budget

The purpose of this appendix is to assess an estimate of the main costs undertaken throughout the development of this Bachelor's Thesis.

The main expenses that are taken into account are the personal expenses, Software expenses and hardware as no experimental procedures were undertaken.

Personal expenses include travel and subsistence allowance expenses and the salary.

- Transportation costs are calculated from more than 60 displacements priced at 6€ per trip. Therefore the transportation expenses add up to 240€.
- Subsistence allowance expenses are calculated from more than 60 meals each priced at 5 €. The total subsistence allowance expenses are 300€.
- The salary expenses of roughly 8 months of nearly complete dedicated work to the research performed. The regular salary agreed by the "XVII Convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos" for " Nivel 2. Diplomados y titulados 1.er ciclo universitario. Jefe Superior" is 1.253,16€×14 payments or annually 17.544,24€[7]. 6 months mean 50% of the total amount that add up to 8.772,12€.

Considering that the only software expense is the student MATLAB license as no other commercialized software has been used, then the total software expenses add up to 69 \$ which are 53,56€.

The hardware expenses are based on the use of the laptop. It has consumed 6 months of the total lifetime of the laptop which has been estimated to be 6 years. Therefore, as the laptop price is 1.200€, the total hardware costs add up to 100 €.

Personal expenses	
Transportation	240€
Subsistence allowance	300€
Salary expenses	8.772, 12€
Software expenses	53, 56€
Hardware expenses	100 €
Total costs	9.465, 68€

Table A.1: Project costs

Appendix B

Nomenclature

- \dot{m} : mass flow rate
- α : mass to power ratio
- ν : collisional frequency
- Φ : electrostatic potential
- χ : Hall parameter
- Γ : Mass flux
- ρ : plasma density
- $(\)_*$: reference quantity for nondimensionalization of $(\)$
- χ^* : reference hall parameter
- Γ_0 : Constant mass flux
- μ_0 : magnetic permeability
- ω_{ce} : cyclotron frequency
- λ_D : Debye Length
- ν_e : collisional frequency
- Γ_i : Ionic flux
- ε_o : vacuum electric permittivity
- $\eta_p \equiv \eta_t$: Thruster or propulsive efficiency
- ν_w : Wall loss frequency
- Δt : time duration
- ΔV : total velocity increment

- β : plasma parameter
- $\sigma_{\parallel} \equiv \sigma$: Parallel conductivity
- σ_{\perp} : Perpendicular conductivity
- σ_H : Hall conductivity
- Rm : Magnetic Reynolds number
- V_{ch} : characteristic velocity
- B : Magnetic field
- c_A : Alfven speed
- c_p : heat capacity
- c_s : sonic speed
- d : electrode distance
- E : Electric field
- e : electron charge
- F_{EM} : electromagnetic force
- F_r : radial force
- F_z : axial force
- g_0 : The standard acceleration due to gravity
- H : complete equivalent field
- I_d : Current delivered
- I_{sp} : specific impulse
- I_{sp}^* : optimum specific impulse
- j : Current density
- k_B : Boltzmann constant
- M : Mach Number
- M_0 : Vehicle Mass
- m_0 : initial mass
- m_1 : final mass
- m_i : ion mass

- M_P : Propellant Mass
- M_{PL} : Payload Mass
- M_{PPS} : Powerplant Mass
- M_{STR} : Structural Mass
- n_e : electrons density
- n_i : ions density
- n_n : neutrals density
- P : momentum
- P_a : power available
- P_d : Power delivered
- p_e : electrons pressure
- p_E : electrostatic pressure
- P_{ei} : electron-ion collisional momentum exchange
- p_i : ions pressure
- $p_{magnetic}$: magnetic pressure
- p_n : neutrals pressure
- P_{req} : power required
- P_T : thrust power
- Q_{cx} : charge exchange collisional rate
- Q_{ei} electron ion collisional rate
- Q_{en} : electron neutron collisional rate
- Q_{ion} ionization rate
- r_a : anode radius
- r_c : cathode radius
- R_e : electrons collisional momentum loss
- R_i : ions collisional momentum loss
- R_n : neutrals collisional momentum loss
- Rm^* : reference magnetic Reynolds number

- S_i : ionic generation
- S_w : loss of ionic flux at the chamber walls
- T : Thrust impelled
- T_c : chamber temperature
- T_e : Electron temperature
- u : velocity
- u_e : electrons velocity
- u_E : exhaust speed
- u_i : ions velocity
- u_n : neutrals velocity
- V : electrostatic potential bias
- V_d : Voltage delivered
- V_{req} : voltage required
- AC: direct current
- AF-MPDT: Applied-field Magnetoplasmdynamic Thruster
- DC: direct current
- EP: Electric Propulsion
- IRS: *Institut für Raumfahrtsysteme*
- KeRC: *Keldish Research Center*
- m: mass of the rocket
- MPDT: Magnetoplasmdynamic Thruster
- SF-MPDT: Self-field Magnetoplasmdynamic Thruster

Appendix C

Matlab Codes

This appendix shows the codes implemented for both models: the Acceleration Model and the Complete Axial Model.

Although it has been described before, the main codes is detailed hereinafter.

ACCELERATION MODEL

1. ACCELERATION_MODEL: Main script.
2. Bisection_method_E_nondim_b_u: Iteration script function.
3. Rm_E: Script for the obtention of nondimensional performance maps.

COMPLETE AXIAL MODEL

1. COMPLETE_AXIAL_MODEL: Main script.
2. event_function_subsonic: Stopping function for the subsonic region.
3. event_function_supersonic: Stopping function for the supersonic region.
4. f_model: function for the calculation of the derivatives needed for the obtention of the Jacobian matrix.
5. com_model_ode: function defining the final mathematical model.

ACCELERATION_MODEL

```
clc
close all
clear all
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% GLOBAL CONSTANTS %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
muo=4*pi*10^(-7);
au=1.66053892*10^(-27);
e=1.60217657*10^(-19);
me=9.10938291*10^(-31);
g=9.80665;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ENGINE PARAMETERS (required for each engine) %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% These are the parameters required by the user of the model

% Two main parameters we will control.
Id=23000; % Current intensity [A]
mdot=0.006; % Mass flow rate [kg/s]

% Geometry of the engine
Rc=0.01; % Cathode radius [m]
Ra=0.05; % Anode radius [m]
Lz=0.1; % Channel length [m]

% Propellant
m_propellant=39.948; % Propellant mass un au units
mi=m_propellant*au; % Actual mass of the propellant

% Approximations for the collisional frequency
Te=5; % Temperature of the electrons [eV]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% GEOMETRY %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This part of the code calculates the remaining geometrical parameters to
% know.
Ly=pi*(Ra+Rc); %% [m]
Lr=Ra-Rc; %% [m]
A=Ly*Lr %% [m^2]
%% PARAMETERS
% Calculates the relevant parameters of the engine using direct formulas of
% the axial model.
Bo=Id*muo/Ly % Azimutal (y) magnetic field [T]
Go=mdot/A % Mass flux [kg/s·m^2]
uE=Bo^2/(2*muo*Go) % Exit velocity [m/s]
Isp=uE/g % Specific Impulse [s]
F=Id^2*muo*Lr/(2*Ly) % Thrust [N]
F_ln=muo*Id^2*log(Ra/Rc)/(4*pi) % Thrust calculated by Maecker's law [N]
nE=Go/(mi*uE) % Number of electrons
lnLAM_E=9+0.5*(log(((10^18)/nE)*(Te)^3)) % Actual lnLAMBDA at the exit.
nue=(nE/(10^18))*(1/Te)^(3/2)*lnLAM_E*2.9*10^(6) % Collisional frequency
Q_ei=nue^2/nE % Collisional rate
SmII=e^2*nE/(me*nue) % Parallel conductivity
wco=e*Bo/me % Cyclotron frequency
Smpp=SmII*nue^2/(wco^2+nue^2) % Perpendicular conductivity
Rmo=SmII*muo*Lz*uE % Characteristic magnetic reynolds number
uEBo=uE*Bo % Induced E field
Puse=F*uE*0.5 % Power of the jet [W]

% Function that evaluates the engine performance at that reynolds
[Eopt,b,zn]=Bisection_method_E_nondim_b_u(Rmo);
% Dimensionalization of the parameter and results of the model
B=b*Bo;
un=1-b.^2;
u=un*uE;
uB=u.*B;
ub=un.*b;
z=zn*Lz;
s=size(z,1);
Eplot=linspace(Eopt,Eopt,s);
E=Eopt*uE*Bo
Vd=E*Lr
Ez=wco.*b.*Eplot'./nue;
```

```

j=SmII*(E-uB);
Eta_p=uE*Bo/(4*E)
PQN=Lr*E*Id

set(0,'DefaultAxesFontSize',22)
set(0,'DefaultAxesFontName','Vijaya')
set(0,'DefaultAxesLineStyleOrder','-');
set(0,'DefaultAxesColorOrder',[0.0 0.0 0.0; 0.4 0.4 0.4; 0.6 0.6 0.6]);

plot(zn,Eplot,'k',zn,b,'r',zn,un,'b',zn,ub,'m','LineWidth',2)
axis tight
set(gcf,'Units','centimeters');
affigurePosition = [10 6 18 12]; % [pos_x pos_y width_x width_y]
set(gcf,'Position',affigurePosition); % [left bottom width height]
set(gcf,'PaperPositionMode','auto');
set(gca,'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 20;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 1; % width of the line of the axes

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
'XLim', [0 1], ... limits for the x-axis
'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties

xlab = xlabel('$\bar{z}$ [-]');
set(xlab,'interpreter','Latex','FontSize',20)
leyenda= legend('$\bar{Ex}$','b','u','$u \times b$');
set(leyenda,'interpreter','Latex','FontSize',14,'LineWidth',2,'Location','North')
ylab = ylabel('$\bar{E}$ [-]');
set(ylab,'interpreter','Latex','FontSize',20)

figure
set(0,'DefaultAxesFontSize',12)

set(gcf,'Units','centimeters');
affigurePosition = [10 6 18 12]; % [pos_x pos_y width_x width_y]
set(gcf,'Position',affigurePosition); % [left bottom width height]
set(gcf,'PaperPositionMode','auto');
set(gca,'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 20;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list

```



```

strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 1; % width of the line of the axes

subplot(2,2,1)
plot(z,B,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
%fonts properties
axis tight
xlab = xlabel('z [ m ]');
set(xlab,'interpreter','Latex','FontSize',20)
ylab = ylabel('$B_{y}$ [ T ]');
set(ylab,'interpreter','Latex','FontSize',14)

axis tight

subplot(2,2,2)
plot(z,u,'LineWidth',2)
axis tight

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]

```

```

'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties

xlab = xlabel('z [ m ]');
set(xlab,'interpreter','Latex','FontSize',20)
ylab = ylabel('$u_z$ [ m s^{-1}]$');
set(ylab,'interpreter','Latex','FontSize',14)

subplot(2,2,3)
plot(z,uB,'LineWidth',2)
axis tight
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
xlab = xlabel('z [ m ]');
set(xlab,'interpreter','Latex','FontSize',20)
ylab = ylabel('$u_z$ \times B_y [ V m^{-1}]$');
set(ylab,'interpreter','Latex','FontSize',14)

subplot(2,2,4)
plot(z,j,'LineWidth',2)
axis tight
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis

```

```

... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
xlab = xlabel('z [ m ]');
set(xlab,'interpreter','Latex','FontSize',20)
ylab = ylabel('$j_x[ A m^{-2} ]$');
set(ylab,'interpreter','Latex','FontSize',14)

```

Bisection_method_E_nondim_b_u

```
function [Eopt,b3,z3]=Bisection_method_E_nondim_b_u(Rm)
    if Rm<0
        Eopt=0
        b3=0
        z3=0
    else
        Emin=2/3^(3/2);
        Emax=100000000;
        ERRORmax=10^(-3);
        ERROR=1;
        counter=1;
        while ERROR>ERRORmax
            counter=1+counter;
            b0 = 1;
            zspan = [0 3];
            Emid=(Emin+Emax)/2;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            Popt=[Emin Rm];
            options = odeset('Events',@event_function);

            [z1,b1,zEmin,bEmin] = ode45(@myode,zspan,b0,options,Popt);

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAX %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            Popt=[Emax Rm];
            options = odeset('Events',@event_function);

            [z2,b2,zEmax,bEmax] = ode45(@myode,zspan,b0,options,Popt);

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MID %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            Popt=[Emid Rm];
            options = odeset('Events',@event_function);

            [z3,b3,zEmid,bEmid] = ode45(@myode,zspan,b0,options,Popt);

            if zEmid>1
                Emin=Emid;
            elseif zEmid<1
                Emax=Emid;
            end
            if counter>10 && Emid<((2/3^(3/2))+0.00001)
                ERROR=ERRORmax-0.00001;
            else
                ERROR=abs(zEmid-1);
            end
        end
        Eopt=Emid
        ERROR
    end
end

function dbdz = myode(z,b,Popt)
    Rm=Popt(2);
    E=Popt(1);
    dbdz=-Rm*(E-b*(1-b^2));
end

function [value,isterminal,direction] = event_function(z,b,Popt)
    value = b; % when value = 0, an event is triggered
    isterminal = 1; % terminate after the first event
    direction = 0; % get all the zeros
end
```

Rm_E

```
function E_Rm=Rm_E
clear all; close all; clc
set(0, 'DefaultAxesColorOrder', [0.0 0.0 0.0; 0.4 0.4 0.4; 0.6 0.6 0.6]);
set(0, 'DefaultAxesLineStyleOrder', '-|--|:');
P=50;
Rm=0.001*linspace(1,10,P);
Rm=[Rm 0.015 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.125 0.15 0.175 0.2 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.5 3 3.5 4 4.5 5 7 8 9 10 20 30 40 50 60 70 80 90 100 150 500 1000];
J=size(Rm,2);

for j=1:J
    Emin=2/3^(3/2);
    Emax=10000;
    ERROR=10^-3;
    while ERROR>10^(-9)
        b0 = 1;
        zspan = [0 3];
        Emid=(Emin+Emax)/2;
        %%%%%%%%% MIN %%%%%%%%%
        Popt=[Emin Rm(j)];
        options = odeset('Events',@event function);
        [z1,b1,zEmin,bEmin] = ode45(@myode,zspan,b0,options,Popt);
        zEmin; % this is the time value where the event occurred
        bEmin; % this is the value of Ca where the event occurred
        %%%%%%%%% MAX %%%%%%%%%
        Popt=[Emax Rm(j)];
        options = odeset('Events',@event function);
        [z2,b2,zEmax,bEmax] = ode45(@myode,zspan,b0,options,Popt);
        zEmax; % this is the time value where the event occurred
        bEmax; % this is the value of Ca where the event occurred
        %%%%%%%%% MID %%%%%%%%%
        Popt=[Emid Rm(j)];
        options = odeset('Events',@event function);
        [z3,b3,zEmid,bEmid] = ode45(@myode,zspan,b0,options,Popt);
        zEmid; % this is the time value where the event occurred
        bEmid; % this is the value of Ca where the event occurred
        if zEmid>1
            Emin=Emid;
        elseif zEmid<1
            Emax=Emid;
        end
        ERROR=abs(zEmid-1);
    end
    Eopt(j)=Emid;
end
Eopt
Rm
%%
figure
set(gcf, 'Units', 'centimeters');
affFigurePosition = [10 6 15 10];
set(gcf, 'Position', affFigurePosition);
set(gcf, 'PaperPositionMode', 'auto');

loglog(Rm,Eopt,'LineWidth',2)
hold on
set(gca, 'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
fontSize = 20;
strFontUnit = 'points';
strFontName = 'Times';
strFontWeight = 'normal';
strFontAngle = 'normal';
strInterpreter = 'latex';
fLineWidth = 1;

set(gca, ...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
```

```

... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
'XLim', [0 1], ... limits for the x-axis
'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes

loglog(Rm,Rm.^(-1),'-k','LineWidth',1)
loglog([0.001 1000],[ (2/(3^(3/2))) (2/(3^(3/2))) ],'-m','LineWidth',1)
loglog(Rm,(2/(3^(3/2)))+Rm.^(-1),'--r','LineWidth',2)
xlab = xlabel('$Rm$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',14)
ylab = ylabel('$\bar{E}$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',14)
t = title('$\bar{E}$ (Rm)$');
set(t,'interpreter','Latex','FontSize',14)
leyenda= legend('Numerical solution','Asymptote $\frac{1}{Rm}$','$\bar{E}_{\min}=\frac{2}{3^{3/2}}\approx 0.3849$', '$\bar{E} = \frac{2}{3^{3/2}} + \frac{1}{Rm}$','Location','NorthEast')
set(leyenda,'interpreter','Latex','FontSize',14)
xlim([10^(-3) 10^3])
ylim([10^(-1) 10^3])
E_Rm=[Eopt Rm]
%%
figure
set(gcf, 'Units', 'centimeters');
affFigurePosition = [10 6 18 12]; % [pos_x pos_y width_x width_y]
set(gcf, 'Position', affFigurePosition); % [left bottom width height]
set(gcf, 'PaperPositionMode', 'auto');

Error=((2/(3^(3/2)))+Rm.^(-1))-Eopt)./Eopt
semilogx(Rm,Error,'LineWidth',2)
set(gca, 'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 20;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 1; % width of the line of the axes

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]

```

```

'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
'XLim', [0 1], ... limits for the x-axis
'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes

xlab = xlabel('Rm [ - ]');
set(xlab, 'interpreter', 'Latex', 'FontSize', 14)
ylab = ylabel('$Error [\bar{E}]$');
set(ylab, 'interpreter', 'Latex', 'FontSize', 14)
t = title('$\frac{E_{approx}-E_{numerical}}{E_{numerical}} (Rm)$');
set(t, 'interpreter', 'Latex', 'FontSize', 14)
xlim([10^-3 10^3])
%%
figure

set(gcf, 'Units', 'centimeters');
affFigurePosition = [10 6 18 12]; % [pos_x pos_y width_x width_y]
set(gcf, 'Position', affFigurePosition); % [left bottom width height]
set(gcf, 'PaperPositionMode', 'auto');

semilogx(Rm, 1./(4*Eopt), 'LineWidth', 2)
hold on
set(gca, 'Units', 'normalized', 'Position', [0.15 0.2 0.75 0.7]);
iFontSize = 20;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 1; % width of the line of the axes

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1', '0', '1'}, ...
... 'YTickLabel', {'-1', '0', '1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
'XLim', [0 1], ... limits for the x-axis
'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
semilogx([0.001 1000], [(3^(3/2)/8)) ((3^(3/2)/8))], '-.m', 'LineWidth', 1)
semilogx(Rm, (3^(3/2)/4)*Rm./(2*Rm+3^(3/2)), '--r', 'LineWidth', 2)
xlab = xlabel('$Rm$ [ - ]');
set(xlab, 'interpreter', 'Latex', 'FontSize', 20)

```

```

ylab = ylabel('$\eta_p$ [ - ]');
set(ylab, 'interpreter', 'Latex', 'FontSize', 20)
t = title('$\eta_p(Rm)$');
set(t, 'interpreter', 'Latex', 'FontSize', 20)
leyenda= legend('Numerical solution', '$\eta_{p_{\max}}=\frac{3^{\{3/2\}}\{8\}\approx0.65$','$\eta_p = \frac{3^{\{3/2\}}\{4\}\frac{Rm}{2Rm+3^{\{3/2\}}}$', 'Location', 'NorthEast')
set(leyenda, 'interpreter', 'Latex', 'FontSize', 14)
xlim([10^(-3) 10^3])
ylim([10^(-3) 1])

end

function dbdz = myode(z,b,Popt)
Rm=Popt(2);
E=Popt(1);
dbdz=-Rm*(E-b*(1-b^2));
end

function [value,isterminal,direction] = event_function(z,b,Popt)
value = b; % when value = 0, an event is triggered
isterminal = 1; % terminate after the first event
direction = 0; % get all the zeros
end

```


COMPLETE_AXIAL_MODEL

```
clc
close all
clear all
format long

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                REFERENCE QUANTITIES                                %
%                                AND                                              %
%                                PARAMETERS CALCULATION                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%CONSTANTS
g=9.80665
e=1.6021765e-19
me=9.11e-31
umass=1.660539e-27
mi=39.948*umass
muo=4*pi*10^(-7)
epso=8.854e-12
kb=1.3806488e-23
%%%GEOMETRY
rc=0.01
ra=0.05
Te=5*e
Gamma=0.7958
Lz=0.1
Lx=ra-rc
Ly=pi*(ra+rc)
%OTHER PARAMETERS
Gammaref=Gamma
Tref=Te
Lref=Lz
%DIMENSIONAL ANALYSIS
uref=sqrt(Tref/mi)
nref=Gammaref/(mi*uref)
nuref=uref/Lref
Qref=nuref/nref
Eref=Tref/(e*Lref)
Bref=Eref/uref
jref=Bref/(muo*Lref)
sigmaref=e^2/(mi*Qref)
Swref=nref*nuref
chiref=Lref/(nref*uref^2)
Rm=sigmatref*muo*uref*Lref
%%% CONSTANTS FOR COLLISIONAL PROCESSES
Eion=15.75*e;
Sion=2.8e-20;
Sen=15e-20;
K2=10.5e-10;
K1=1.67e-10
lnLAM=9;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Qion=sqrt((8*Te)/(pi*me))*Sion*(1+((Te*Eion)/(Te+Eion)^2))*exp(-Eion/Te)
Qen=sqrt((8*Te)/(pi*me))*Sen
Qcx=uref*(K2-K1*log10(uref))^2
Qei=(Te/e)^(-3/2)*lnLAM*2.9e-12
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
qion=Qion/Qref
qen=Qen/Qref*(me/mi)
qcx=Qcx/Qref
qei=Qei/Qref*(me/mi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                SOLUTION OF THE MODEL                                %
%                                CONDITIONS AT S THE SONIC POINT                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % Te=2eV
% zS=0;                %% z at sonic point
% uiS=1;               %% ui at sonic point
% giS=0.7;            %% gi at sonic point
% bS=13;              %% b at sonic point
% unS=0.75;           %% un at sonic point

% % Te=3eV
```

```

% zS=0;                %% z at sonic point
% uiS=1;               %% ui at sonic point
% giS=0.625;          %% gi at sonic point
% bS=13.8;            %% b at sonic point
% unS=0.66;           %% un at sonic point

% % Te=4eV
% zS=0;                %% z at sonic point
% uiS=1;               %% ui at sonic point
% giS=0.87;           %% gi at sonic point
% bS=11.9;            %% b at sonic point
% unS=0.56245;        %% un at sonic point

% % Te=5eV
zS=0;                %% z at sonic point
uiS=1;               %% ui at sonic point
giS=0.84;            %% gi at sonic point
bS=11.9;             %% b at sonic point
unS=0.613;           %% un at sonic point

niS=giS/uiS;

S0=[zS uiS niS bS unS]; %% Vector of conditions at S

nnS=(1-giS)/unS;      %% nn at sonic point
sigmaS=niS/(qen*nnS+qei*niS);
sw=0;
exS=((uiS*niS*nnS*(uiS-unS)*(qcX+qion)+(niS*nnS*qion-sw))/(uiS*sigmaS*bS))+uiS*bS

Popt=[exS qion qen qcX qei sw Rm]; %% Vector of parameters for ode45.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATION OF THE EIGEN PROBLEM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FOR INITIAL CONDITIONS FOR BOTH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% REGIONS, SUBSONIC AND SUPERSONIC.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

yS=[zS uiS niS bS unS];
ySn=[1 uiS niS bS unS];
delta=0.001;
DELTA=delta*[1 1 1 1 1].*ySn;
YSU=[zS+DELTA(1) uiS niS bS unS;zS uiS+DELTA(2) niS bS unS;zS uiS niS+DELTA(3) bS unS;zS uiS niS
bS+DELTA(4) unS; zS uiS niS bS unS+DELTA(5)];
YSL=[zS-DELTA(1) uiS niS bS unS;zS uiS-DELTA(2) niS bS unS;zS uiS niS-DELTA(3) bS unS;zS uiS niS bS-
DELTA(4) unS; zS uiS niS bS unS-DELTA(5)];

%% GENERATING VECTOR OF DERIVATIVES
for k=1:size(YSU,2)
    fSU(:,k)=f_model(YSU(k,:),Popt);
    fSL(:,k)=f_model(YSL(k,:),Popt);
end
%% GENERATING JACOBEAN MATRIX FROM VECTOR OF DERIVATIVES AND DELTA VECTOR
%% (linear interpolation of slopes)
for r=1:size(fSU,1)
    for s=1:size(fSU,2)
        JfyS(r,s)=(fSU(r,s)-fSL(r,s))/(2*DELTA(r));
    end
end
JfyS
[V,D]=eig(JfyS)                %% Solving for the eigenvectors and eigenvalues

[row,col]=find(D==max(max(D))) %% Select the index of the highest eigenvalue
V1=V(:,row)                    %% Select the eigenvector from the column of the prev.index

%% The values of V1 are increments of the variables. These increments must
%% be normalized with respect to S0. Therefore the maximum increment will
%% be a 5% of the initial condition.

V2(3:5)=V1(3:5)'/S0(3:5)      %% V2 is an additional vector to compare derivatives
V2(1:2)=V1(1:2)
delta=0.05/max(abs(V2))
VA=delta*V1                    %% Max of derivatives is 5% of the initial condition

%% In order to define the initial conditions, it is very important to take
%% into account that the sign of dzdxi is uncertain. Therefore the sign
%% command is used.

S0sS=S0-sign(VA(1))*VA'        %% INITIAL CONDITION FOR THE SUBSONIC REGION
S0SS=S0+sign(VA(1))*VA'        %% INITIAL CONDITION FOR THE SUPERSONIC REGION

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SUPERSONIC REGION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    chispanSS=[0 10000];
    options = odeset('Events',@event_function_supersonic,'AbsTol',[1e-04 1e-04 1e-04 1e-04 1e-
04]);
    PoptSS=[exS qion qen qcx qei sw Rm];
    [chiSS,ySS] = ode45(@com_model_ode,chispanSS,S0SS,options,PoptSS);

    zSS=ySS(:,1);
    uiSS=ySS(:,2);
    niSS=ySS(:,3);
    bSS=ySS(:,4);
    unSS=ySS(:,5);
    chiSS=chiSS;
    nnSS=(1-uiSS.*niSS)./unSS;
    sigmaSS=(niSS./(qei*niSS+qen*nnSS));
    GSS=uiSS.*bSS.*sigmaSS.*(exS-uiSS.*bSS)-niSS.*nnSS*qion;
    GSSp=uiSS.*bSS.*sigmaSS.*(exS-uiSS.*bSS);
    GSSn=-niSS.*nnSS*qion;
    dbSS=-niSS.*(uiSS.^2-1).*(sigmaSS).*(exS-uiSS.*bSS);

    set(0,'DefaultAxesFontSize',14);
    zend=zSS(end);
    bend=bSS(end);
    Gend=GSS(end);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SUBSONIC REGION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    chispansS=[0 10000];
    options = odeset('Events',@event_function_subsonic,'AbsTol',[1e-04 1e-04 1e-04 1e-04 1e-
04]);
    PoptS=[exS qion qen qcx qei sw Rm];
    [chisS,ysS] = ode45(@com_model_ode,chispansS,S0sS,options,PoptS);
    zsS=ysS(:,1);
    uisS=ysS(:,2);
    nisS=ysS(:,3);
    bsS=ysS(:,4);
    unsS=ysS(:,5);

    nnsS=(1-uisS.*nisS)./unsS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% RESULTS FOR EACH REGION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

set(0,'DefaultAxesFontSize',12);

set(gcf, 'Units', 'centimeters','DefaultAxesColorOrder',[0 0 0]);
affFigurePosition = [10 6 20 11]; % [pos_x pos_y width_x width_y]
set(gcf, 'Position', affFigurePosition); % [left bottom width height]
set(gcf, 'PaperPositionMode', 'auto');
set(gca, 'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 9;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 0.5; % width of the line of the axes

subplot(2,3,1)
plot(zSS,bSS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis

```

```

... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('b [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)
axis tight

subplot(2,3,2)
plot(zSS,uiSS,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight

ylab = ylabel('u$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)
axis tight

subplot(2,3,3)

```

```

plot(zSS,niSS,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight

ylab = ylabel('n$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,4)
plot(zSS,unSS,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels

```

```

'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
% xlabel('FontName','Cambria','FontSize',14)
% ylabel('B_y [T]','FontName','Cambria','FontSize',14)

% xlab = xlabel('z [ m ]');
% set(xlab,'interpreter','Latex','FontSize',20)
% ylab = ylabel('$B_{y}$ [ T ]');
% set(ylab,'interpreter','Latex','FontSize',14)

ylab = ylabel('u_n$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,5)
plot(zSS,unSS.*nnSS,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
'XTick', 0:0.1:100, ... ticks of x axis
'YTick', 0:1:10, ... ticks of y axis
'XTickLabel', {'-1','0','1'}, ...
'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
% xlabel('FontName','Cambria','FontSize',14)
% ylabel('B_y [T]','FontName','Cambria','FontSize',14)

% xlab = xlabel('z [ m ]');
% set(xlab,'interpreter','Latex','FontSize',20)
% ylab = ylabel('$B_{y}$ [ T ]');
% set(ylab,'interpreter','Latex','FontSize',14)

ylab = ylabel('G_n$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,6)
plot(zSS,uiSS.*niSS,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO

```

```

...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
% xlabel('FontName','Cambria','FontSize',14)
% ylabel('B_y [T]','FontName','Cambria','FontSize',14)

ylab = ylabel('G$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

ZSS=zSS*Lref;
UiSS=uiSS*uref;
NiSS=niSS*nref;
BSS=bSS*Bref;
UnSS=unSS*uref;
ExS=exS*Eref;
ChiSS=chiSS*chiref;
NnSS=nnSS*nref;

figure

set(gcf, 'Units', 'centimeters','DefaultAxesColorOrder',[0 0 0]);
affFigurePosition = [10 6 20 11]; % [pos_x pos_y width_x width_y]
set(gcf, 'Position', affFigurePosition); % [left bottom width height]
set(gcf, 'PaperPositionMode', 'auto');
set(gca, 'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 9;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = .5; % width of the line of the axes

subplot(2,3,1)
plot(zsS,bsS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]

```

```

...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight

ylab = ylabel('b [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,2)
plot(zsS,uisS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increase direction [{normal} | reverse]
'YDir', 'normal', ... axis increase direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('u$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

```



```

subplot(2,3,3)
plot(zsS,nisS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('n$ i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,4)
plot(zsS,unsS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels

```

```

'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('u$_n$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,5)
plot(zsS,unsS.*nnsS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('G$_n$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,6)
plot(zsS,uisS.*nisS,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]

```

```

'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increaseement direction [{normal} | reverse]
'YDir', 'normal', ... axis increaseement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('G$_i$ [ - ]');
set(ylab, 'interpreter', 'Latex', 'FontSize', 16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab, 'interpreter', 'Latex', 'FontSize', 16)

ZsS=zsS*Lref;
UisS=uisS*uref;
NisS=nisS*nref;
BsS=bsS*Bref;
UnsS=unsS*uref;
ChisS=chisS*chiref;
NnsS=nnsS*nref;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% COMPLETE SOLUTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xsS=size(zsS,1)
xSS=size(zsS,1)
ysS=size(uisS,1)
ySS=size(uiSS,1)
for i=1:xsS
    z(i)=zsS(xsS+1-i)
end

for i=1:xSS
    z(i+xsS)=zSS(i)
end
z=z-zsS(end)

for i=1:ysS
    ui(i)=uisS(ysS+1-i);
    ni(i)=nisS(ysS+1-i);
    b(i)=bsS(ysS+1-i);
    un(i)=unsS(ysS+1-i);
end

for i=1:ySS
    ui(i+ysS)=uiSS(i);
    ni(i+ysS)=niSS(i);
    b(i+ysS)=bSS(i);
    un(i+ysS)=unSS(i);
end
nn=(1-ui.*ni)./un
Z=z*Lref;
Ui=uref*ui;
Ni=nref*ni;
B=Bref*b;
Un=uref*un;
Nn=(uref*nref-Ui.*Ni)./Un;
Nn2=nn*nref
Error=Nn-Nn2
Sn=sgmaref*(ni./(ni*qi+(Nn/nref)*qen));

ExS/(ui(end)*uref*bsS(end)*Bref)
uE=(Ui(end)*Ni(end)+Un(end)*Nn(end))/(Ni(end)+Nn(end))
Isp=uE/g
Bo=B(1)
A=Lx*Ly
Lzobt=z(end)*Lref
mdot=mi*(Ni(end)+Nn(end))*uE*A
F=uE*mdot
Puse=F*uE/2
ExS

```

```

Vd=ExS*Lx
Id=Bo*Ly/muo
Pd=Vd*Id
ETA=Puse/Pd
no=Ni(1)
lmdD=sqrt((epso*Te)/(e^2*no))
%%% Backplate Conditions
GiA=ui(1)*ni(1)
UnA=Un(1)
uiA=ui(1)
UiA=Ui(1)
%%% Exit Conditions
GiE=ui(end)*ni(end)

figure
set(0,'DefaultAxesFontSize',12);
set(gcf,'Units','centimeters','DefaultAxesColorOrder',[0 0 0]);
affFigurePosition = [10 6 20 11]; % [pos_x pos_y width_x width_y]
set(gcf,'Position',affFigurePosition); % [left bottom width height]
set(gcf,'PaperPositionMode','auto');
set(gca,'Units','normalized','Position',[0.15 0.2 0.75 0.7]);
iFontSize = 9;
strFontUnit = 'points'; % [{points} | normalized | inches | centimeters | pixels]
strFontName = 'Times'; % [Times | Courier | ] TODO complete the list
strFontWeight = 'normal'; % [light | {normal} | demi | bold]
strFontAngle = 'normal'; % [{normal} | italic | oblique] ps: only for axes
strInterpreter = 'latex'; % [{tex} | latex]
fLineWidth = 0.5; % width of the line of the axes

subplot(2,3,1)
plot(z,b,'LineWidth',2)
set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
ylab = ylabel('b [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)
axis tight

subplot(2,3,2)
plot(z,ui,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO

```

```

...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
ylab = ylabel('u$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)
axis tight

subplot(2,3,3)
plot(z,ni,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1','0','1'}, ...
...'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties

```

```

axis tight
ylab = ylabel('n$_i$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,4)
plot(z,un,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
... 'XLim', [0 1], ... limits for the x-axis
... 'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('u$_n$ [ - ]');
set(ylab,'interpreter','Latex','FontSize',16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab,'interpreter','Latex','FontSize',16)

subplot(2,3,5)
plot(z,un.*nn,'LineWidth',2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
... 'XTick', 0:0.1:100, ... ticks of x axis
... 'YTick', 0:1:10, ... ticks of y axis
... 'XTickLabel', {'-1','0','1'}, ...
... 'YTickLabel', {'-1','0','1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]

```

```

'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('G$_n$ [ - ]');
set(ylab, 'interpreter', 'Latex', 'FontSize', 16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab, 'interpreter', 'Latex', 'FontSize', 16)

subplot(2,3,6)
plot(z, ui.*ni, 'LineWidth', 2)

set(gca, ...
... 'Position', [1 1 20 10], ... TODO
... 'OuterPosition', [1 1 20 10], ... TODO
...
'XGrid', 'off', ... [on | {off}]
'YGrid', 'off', ... [on | {off}]
'GridLineStyle', ':', ... [- | -- | {:} | -. | none]
'XMinorGrid', 'off', ... [on | {off}]
'YMinorGrid', 'off', ... [on | {off}]
'MinorGridLineStyle', ':', ... [- | -- | {:} | -. | none]
...
...'XTick', 0:0.1:100, ... ticks of x axis
...'YTick', 0:1:10, ... ticks of y axis
...'XTickLabel', {'-1', '0', '1'}, ...
...'YTickLabel', {'-1', '0', '1'}, ...
'XMinorTick', 'off', ... [on | {off}]
'YMinorTick', 'off', ... [on | {off}]
'TickDir', 'out', ... [{in} | out] inside or outside (for 2D)
'TickLength', [.01 .01], ... length of the ticks
...
'XColor', [.1 .1 .1], ... color of x axis
'YColor', [.1 .1 .1], ... color of y axis
'XAxisLocation', 'bottom', ... where labels have to be printed [top | {bottom}]
'YAxisLocation', 'left', ... where labels have to be printed [left | {right}]
'XDir', 'normal', ... axis increasement direction [{normal} | reverse]
'YDir', 'normal', ... axis increasement direction [{normal} | reverse]
...'XLim', [0 1], ... limits for the x-axis
...'YLim', [0 1], ... limits for the y-axis
...
'FontName', strFontName, ... kind of fonts of labels
'FontSize', iFontSize, ... size of fonts of labels
'FontUnits', strFontUnit, ... units of the size of fonts
'FontWeight', strFontWeight, ... weight of fonts of labels
'FontAngle', strFontAngle, ... inclination of fonts of labels
...
'LineWidth', fLineWidth); % width of the line of the axes
% fonts properties
axis tight
ylab = ylabel('G$_i$ [ - ]');
set(ylab, 'interpreter', 'Latex', 'FontSize', 16)
xlab = xlabel('$\bar{z}$ [ - ]');
set(xlab, 'interpreter', 'Latex', 'FontSize', 16)

```

event_function_subsonic

```
function [value,isterminal,direction] = event_function_subsonic(z,S,Popt)
% when value is equal to zero, an event is triggered.
% set isterminal to 1 to stop the solver at the first event, or 0 to
% get all the events.
% direction=0 if all zeros are to be computed (the default), +1 if
% only zeros where the event function is increasing, and -1 if only
% zeros where the event function is decreasing.
value(1) = S(2)+1;
value(2) = S(1)-10;
% LIMITATIONS FOR THE DERIVATIVES
% value = S(4)*(S(1)-1)*((S(2)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-
S(2)*S(3))/S(2))*Popt(3))-((1-S(2)*S(3))/S(2))*S(3)*Popt(2))); % when value = 0, an event is
triggered
% value(3) = ((S(2)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-S(2)*S(3))/S(5))*Popt(3))-
S(2)*S(3)*((1-S(2)*S(3))/S(5))*S(2)-S(5))*(Popt(4)+Popt(2))-((1-S(2)*S(3))/S(5))*S(3)*Popt(2)-
Popt(6)));
% value(4) = -1*((S(3)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-S(2)*S(3))/S(5))*Popt(3))-
S(3)*S(3)*((1-S(2)*S(3))/S(5))*S(2)-S(5))*(Popt(4)+Popt(2))-S(2)*S(3)*((1-
S(2)*S(3))/S(5))*S(3)*Popt(2)-Popt(6)));
value(3) = S(5)-0.000001;
% ((1-S(2)*S(3))/S(5))*S(3)*Popt(4)*(S(2)-S(5))-Popt(6)*S(5))/(1-S(2)*S(3));
isterminal(1) = 1; % terminate after the first event
isterminal(2) = 1; % terminate after the first event
isterminal(3) = 1; % terminate after the first event
direction(1) = 0; % get all the zeros
direction(2) = 0; % get all the zeros
direction(3) = 0; % get all the zeros
end
```

event_function_supersonic

```
function [value,isterminal,direction] = event_function_supersonic(z,S,Popt)
% when value is equal to zero, an event is triggered.
% set isterminal to 1 to stop the solver at the first event, or 0 to
% get all the events.
% direction=0 if all zeros are to be computed (the default), +1 if
% only zeros where the event function is increasing, and -1 if only
% zeros where the event function is decreasing.
value(1) = S(4);
value(2) = S(1)-100;
%%% STOPPING CONDITIONS FOR THE DERIVATIVES.
% value = S(4)*(S(1)-1)*((S(2)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-
S(2)*S(3))/S(2))*Popt(3))-((1-S(2)*S(3))/S(2))*S(3)*Popt(2))); % when value = 0, an event is
triggered
% value(3) = ((S(2)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-S(2)*S(3))/S(5))*Popt(3))-
S(2)*S(3)*((1-S(2)*S(3))/S(5))*S(2)-S(5))*(Popt(4)+Popt(2))-((1-S(2)*S(3))/S(5))*S(3)*Popt(2)-
Popt(6)));
% value(4) = -1*((S(3)*S(3)*S(4)*(Popt(1)-S(4)*S(2))/(S(3)*Popt(5)+((1-S(2)*S(3))/S(5))*Popt(3))-
S(3)*S(3)*((1-S(2)*S(3))/S(5))*S(2)-S(5))*(Popt(4)+Popt(2))-S(2)*S(3)*((1-
S(2)*S(3))/S(5))*S(3)*Popt(2)-Popt(6)));
isterminal(1) = 1; % terminate after the first event
isterminal(2) = 1; % terminate after the first event
isterminal(3) = 1; % terminate after the first event
isterminal(4) = 1; % terminate after the first event
direction(1) = 0; % get all the zeros
direction(2) = 0; % get all the zeros
% direction(3) = 0; % get all the zeros
% direction(4) = 0; % get all the zeros
end
```


f_model

```
function dy = f_model(y,Popt)
ex=Popt(1);
qion=Popt(2);
qen=Popt(3);
qcx=Popt(4);
qei=Popt(5);
sw=Popt(6);
Rm=Popt(7);

z=y(1);
ui=y(2);
ni=y(3);
b=y(4);
un=y(5);

gi=ni*ui;
nn=(1-gi)/un;
sigma=ni/(ni*qei+nn*qen);

dz=ni*(ui^2-1);
dui=ui*sigma*b*(ex-ui*b)-ui*ni*nn*(ui-un)*(qcx+qion)-(ni*nn*qion-sw);
dni=-ni*sigma*b*(ex-ui*b)+ni^2*nn*(ui-un)*(qcx+qion)+(ni*ui)*(ni*nn*qion-sw);
db=-dz*sigma*Rm*(ex-ui*b);
dun=dz*(ni*nn*(ui-un)*qcx-un*sw)/(1-gi);

dy=[dz; dui; dni; db; dun]
end
```

com_model_ode

```
function dS = com_model_ode(chi,S,Popt)
ex=Popt(1);
qion=Popt(2);
qen=Popt(3);
qcx=Popt(4);
qei=Popt(5);
sw=Popt(6);
Rm=Popt(7);

z=S(1);
ui=S(2);
ni=S(3);
b=S(4);
un=S(5);

gi=ni*ui;
nn=(1-gi)/un;
sigma=ni/(ni*qei+nn*qen)

dz=ni*(ui^2-1);
dui=ui*sigma*b*(ex-ui*b)-ui*ni*nn*(ui-un)*(qcx+qion)-(ni*nn*qion-sw);
dni=-ni*sigma*b*(ex-ui*b)+ni^2*nn*(ui-un)*(qcx+qion)+(ni*ui)*(ni*nn*qion-sw);
db=-dz*sigma*Rm*(ex-ui*b);
dun=dz*(ni*nn*(ui-un)*qcx-un*sw)/(1-gi);

dS=[dz; dui; dni; db; dun];
[S dS]
chi
end
```